# Practical Limits to Transfer Learning of Neural Network Controllers from Earth to Space Environments

Collins Ogundipe(✉) and Alex Ellery

Department of Mechanical and Aerospace Engineering, Carleton University, 1125 Colonel by Drive, Ottawa K1S 5B6, Canada
collinsogundipe@cmail.carleton.ca

**Abstract.** Given the similarity in form and dynamics between earth-based and space-based robotic manipulators, transfer learning of neural network controllers would naturally be a plausible avenue to address the challenges of limited computation resources onboard the spacecraft (space manipulator). We have introduced a pretrained and learned feedforward neural network for modeling the control error a priori. While the results are encouraging, there are major limitations of neural networks' capability to ensuring the transfer learning of similar earth-based dynamics to space-based dynamics, given that the parameters of contrast are fairly straightforward. To show these limitations, we present a novel approach that is inspired by human motor control. We have explored the adaptability of neural networks as a key feature for robust AI which has traditionally suffered from brittleness. This was demonstrated through a practical problem of transferring a neuro-controller from earth to space. It was discovered that neural networks including deep learning models are still too brittle for general AI. We have developed appropriate neural network models using trajectory data from 7 degrees-of-freedom (7-DOF) Barrett Arm, as representative of an earth-based to space-based manipulator kinematics and dynamics.

**Keywords:** Transfer learning · Neural network · Forward model · Space robotics · Manipulator · Free-flyer

## 1 Introduction

In human level manipulation, various sections of the brain extend into the motor area M1 to supply feedback signals. The parietal cortex, for instance, deals with visual control of hand motions, and it calculates the error between the current cartesian position and the desired cartesian position [1]. To do this, an efference copy of the motor commands is required to produce a feedforward compensation. The efference copy of the motor commands is typically transmitted to an emulator which models the input-output response of the musculoskeletal system. From a biomimetic perspective, it is believed that a hierarchical neural network system in any control architecture can imitate this function of the motor cortex [2]. During human manipulation, the error between the actual motor


M. Bramer and F. Stahl (Eds.): SGAI-AI 2022, LNAI 13652, pp. 3–16, 2022.
https://doi.org/10.1007/978-3-031-21441-7_1

outputs (joint position ($\theta$) and joint velocity ($\dot{\theta}$) evaluated by the proprioceptors) and the commanded motor input (torque $\tau$, from the motor cortex) is fed back as $[\theta^{desired} - \theta]$ having a time delay of 40–60 ms [3]. However, a "forward dynamics model of the musculoskeletal system exists within the spinocerebellum-magnocellular red nucleus system" [3]. This forward model accepts feedback ($\theta$ and $\dot{\theta}$) from the proprioceptors and an afferent copy of the motor command ($\tau$) from the motor cortex. Consequently, the forward model receives motor command $\tau$ as its input and outputs an estimated predictive trajectory $\theta^*$ [3], processing this input-output comparison between the pair ($\tau$ and $\theta^*$) to generate a predicted error $[\theta^{desired} - \theta^*]$ in a much faster manner to minimize the error. The forward model does this prediction/comparison in 10–20 ms, transmitting this to the motor cortex in the process [3]. The sensory effects of the motor command are predicted by this forward model. This type of top-down prediction model is centered on the statistical reproducible model of the causative nature of the world learned via input-output pairs. This can be directly explored with predictive neural networks as forward model by adopting input-output models of deep learning architecture or multivariate regression. In human level interaction, these forward models of the musculoskeletal system have been learned through the initial motor babbling that started from infancy [3]. And the learned models are transferred to adapt to changes in stimuli or environments, given the underlying dynamics remain the same.

This leads to the practical problem we have detailed in this paper, which is the transfer learning from earth-based manipulators to space-based manipulators. In space robotics, there are simply two fundamental changes from earth to space which are accounted for through: (i) the absence of gravity in space, and (ii) the direct substitutions of certain derived parameters which are quantified in numbers and readily available as a modification of the earth-based equivalents. So, essentially, the dynamics of the robotic system remain the same, and necessary environmental variations are readily accounted for. All other space-based environmental factors are known to be negligible as they pertain to the dynamics of space robot's interaction. The environmental disturbance torques (gravity gradient, aerodynamics and magnetic torques) imposed on the robot's spacecraft are very small – within 10e−6 Nm [4]. The primary differentiating characteristics of space robotics from terrestrial robotics is that the robot operates in a microgravity environment. Transfer learning of neural network controller trained as a forward model in a biomimetic approach similar to how human manipulation is carried out should be able to exhibit efficient generalization as typically shown for new data input in most deep learning domain/applications. However, the practical limitation of transfer learning of neural network controllers is the exhibition of lack of general intelligence, as detailed in this paper.

Considerable effort has been put into developing machine learning methods that can learn and improve inverse dynamics model of robotic manipulators [5–8]. Online learning has been the focus in these settings because when considering motions with object interactions, learning one global model becomes very challenging, if not impossible, since the model must be a function of contact and payload signals. To approach the issue of global/dynamic model, learning task-specific (error) models has been proposed in the past [9–12], such that the overall global problem is simplified into two subproblems – (1) finding a task-specific inverse dynamics model and (2) detecting which task

model to use. This permits to iterate the collection of data specific to a task, learn an error model, and then apply the learned model during the required task execution. However, a key difficulty that has been encountered is the computationally efficient learning of models that are data-efficient as possible, such that only few iterations are required while achieving consistent convergence in the error model learning. We seek to address this using predictive feedforward approach, in a pre-learned fashion, by ensuring the transfer learning of earth-based model to space environment. Our take on this is that pre-learned input-output models are computationally efficient compared with analytical models – the latter require exact knowledge of parameters (commonest sources of errors which include payload variation) and require computation time. Learned models reduce computation by storing model in memory, which also ensure a more compliant and reactive robot.

For feedback control to work, errors must exist to invoke the corrective behaviour. This is not the case for feedforward control which does not require errors to work. Forward model implemented in conjunction with feedback control reduces the potential error excursions [13]. Currently, space manipulators are typically teleoperated in space by astronauts or by ground operators and are operated very slowly. This acts as a severe restriction on productivity rates. The incorporation of feedforward controllers, therefore, offers the advantage to robustify and speed up operations as they do not require error excursion to function. In the following, we first described in Sect. 2, the background to the derived parameters relating space-based manipulator's kinematics and dynamics to earth-based environment. In Sect. 3, we detailed a novel predictive feedforward control via a forward model; followed by a complete overview of our learning algorithms and manipulator configuration in Sect. 4. Finally, we evaluate the results of the proposed scheme in Sect. 5 and outlined the practical limitations of transfer learning of the neural network controller. Section 6 detailed the conclusion which exposes problems in neural networks including deep learning as a model of general intelligence.

## 2 Space-Based Kinematics and Dynamics

We must first consider the kinematics and dynamics of a freeflyer-mounted manipulator. The main differentiating characteristics of space robots from terrestrial robots is that terrestrial robots are mounted onto a firm ground; in space, we have no such force or torque reaction cancellation to the movement of manipulator arms. Additionally, the robot operates in a microgravity environment; hence, the kinematics and dynamics of free-flying robotic manipulator deployed in space will take a different approach. In the consideration of a free-flying robotic manipulator mounted on a spacecraft bus having dedicated attitude control, the position kinematics ($p^*$) of the manipulator in connection with inertial space is given by [14, 15]:

$$p^* = r_{c0} + R_0 s_0 + \sum_{i=1}^{n} R_i l_i \tag{1}$$

where $r_{c0}$ is the position of the spacecraft centre of mass with respect to the inertial coordinates; $R_0$ is the attitude of the spacecraft with respect to the inertial coordinates;

$s_0$ is the position vector of the manipulator base with respect to the spacecraft body centre of mass; $R_i$ is the 3-by-3 direction cosine matrix of each link with respect to the base coordinates; $n$ is the number of serial rigid body links; $i$ represents the link number from 0 to $n$; while $l_i$ is the vectoral length of link $i$ from $(x_{i-1}, y_{i-1}, z_{i-1})$ to $(x_i, y_i, z_i)$ (Shown in Fig. 1.)

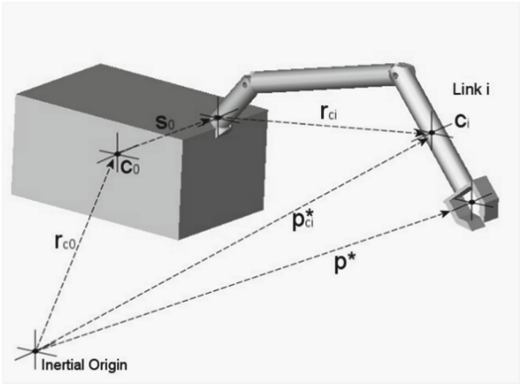


**Fig. 1.** Spacecraft-Manipulator Geometry ($C_0$ represents spacecraft's center of mass; $r_{ci}$ is the distance between the centres of mass of adjacent links with respect to the base coordinates; $C_i$ is the center of mass of link $i$; $p^*_{ci}$ is the position of the link $i$ centre of mass with respect to the inertial coordinates).

For spacecraft bus with dedicated attitude control, $R_0 = I_3$ (identity matrix). The center of mass of the whole system (the robotic manipulator, satellite bus mount, and the payload) is represented by [14, 15]:

$$p^*_{cm} = \frac{\sum_{i=0}^{n+1} m_i p^*_{ci}}{\sum_{i=0}^{n+1} m_i} \tag{2}$$

where $p^*_{cm}$ is the location of the centre of mass of the complete manipulator/spacecraft system with regards to the inertial coordinates; $m_i$ is the mass of each component rigid body links; $n$ is the number of rigid body links; $n = 0$ represents the spacecraft body link; $p^*_{ci}$ is the position of link $i$ centre of mass in reference to the inertial coordinates. Similarly to terrestrial manipulator algorithms in the form of $p_i = R_i l_i$, the equation of the space manipulator for the location of the center of mass of the complete manipulator/spacecraft system with regards to the inertial coordinates ($p^*_{cm}$) has been derived to be [16–18]:

$$\begin{aligned} p^*_{cm} = r_{c0} + \left(1 - \frac{m_0}{m_T}\right) s_0 + \frac{1}{m_T} \sum_{i=1}^{n} R_i \left( \sum_{j=i+1}^{n+1} m_j l_i + m_i r_i \right) .. \\ + \frac{m_{n+1}}{m_T} R_{n+1} r_{n+1} \end{aligned} \tag{3}$$

where $m_0$ is the mass of the spacecraft bus; $m_T$ is the total mass of the system; $m_i$ is the mass of each component rigid body i comprising the system; $r_i$ is the vectorial distance

from the origin of link i to the centre of mass of link i; $n+1$ represents the corresponding notations for the payload link. Equation (3) was separated into three parts: parts related to body 0 (the spacecraft), bodies 1 to $n$ (the manipulator links) and body $n+1$ (for the payload). This then reduces to [18]:

$$p^*_{cm} = r_{c0} + \left(1 - \frac{m_0}{m_T}\right)s_0 + \sum_{i=1}^{n} R_i L_i + \left(\frac{m_{n+1}}{m_T}\right)r_{n+1}$$
$$\text{where } L_i = \frac{1}{m_T}\left(\sum_{j=i+1}^{n+1} m_j l_i + m_i r_i\right) \tag{4}$$

This concludes the location of center of mass of the system with respect to inertial space. It is assumed arbitrarily that the local inertial reference frame initially coincides with the spacecraft bus center of mass, that is, $r_{c0} = 0$, since any point fixed in the interceptor body could be regarded as inertially fixed prior to any robotic maneuver [18]. Having defined $p^*_{cm}$, the term $r_{c0}$ is then substituted into Eq. (1), which gives

$$p^* = p^*_{cm} + s_0 + \sum_{i=1}^{n} R_i l_i - \frac{1}{m_T}\sum_{i=1}^{n+1}\sum_{j=i}^{n+1} m_j r_{ci} \tag{5}$$

This is further simplified into:

$$p^* = p^*_{cm} + s_0 + \sum_{i=1}^{n} R_i l_i - \ldots$$
$$\frac{1}{m_T}\sum_{i=1}^{n+1}\sum_{j=i}^{n+1} m_j (R_i r_i + R_{i-1} s_{i-1}) \tag{6}$$

where $r_{ci} = R_i r_i + R_{i-1} s_{i-1}$ [18]. Similarly, we separate out the three parts associated to the spacecraft mount (body 0), bodies 1*ton* for the manipulator links and body n + 1 for the payload [18]. This gives

$$p^* = p^*_{cm} + \frac{m_0}{m_T} s_0 + \sum_{i=1}^{n} R_i \lambda_i - \frac{m_{n+1}}{m_T} R_{n+1} r_{n+1}$$
$$\text{where } \lambda_i = \frac{1}{m_T}\sum_{j=0}^{i} (m_j l_j - m_i r_i) \tag{7}$$

Accordingly, $\lambda_i$ is referred to as the lumped kinematic parameter for each manipulator link. The Eq. (7) of $p^*$ is an equivalent form to that of the terrestrial-based manipulator of the form $p = \sum_{i=1}^{n} R_i l_i$ with added constants; ($p^*_{cm}$ is constant, and $\lambda_i$ is constant as the lumped kinematic/dynamic parameter, replacing the $l_i$ in terrestrial-based manipulator).

Therefore, the inverse kinematics solution to the space manipulator geometry can be found with little modifications to the terrestrial algorithms.

## 3 Predictive Feedforward Control

Our bio-inspired error-learning approach addresses the need for reactive and adaptive behavior to diverse range of tasks under dynamic environmental conditions. If we could successfully demonstrate this for a terrestrial manipulator, the idea is to incorporate the approach in a free-flyer concept for the removal of space debris of varying sizes; with the aim to offer a solution transferrable from earth to different orbital bands. In effect, we propose here a control scheme that is centered on biomimetic models for predictive forward control in conjunction with traditional feedback control. We believe that bio-inspired forward models could provide solution for adaptive and robust control, which could position robotic manipulators for the complex task of salvaging space debris if the learned model can be successfully transferred to space environment. Adaptivity will be implemented through learning of new forward models to adapt to new situations; robustness is implemented in the form of forward models that provide rapid behavior without relying on error excursions unlike traditional feedback controllers. The superiority of feedforward-feedback control over feedback control only has been clearly demonstrated [13]. Pure feedback control is implausible for reactive manipulation due to substantial delays in sensors' feedback signals. This is like the case biologically where human reaction time is limited to a maximum of about 400–500 ms [19]. Therefore, predictive feedforward strategy is proposed as added measure to correct the robot's trajectory along with the feedback control. In this current study, we have not yet implemented feedback delays into the forward model yet – the work presented here is the first step in building a more comprehensive and sophisticated manipulator control system. The bio-inspired control system should comprise a paired feedforward-feedback system with a learning system that adapts forward models for different scenarios such as time delays and/or payload variations. Hence, the core of this approach is the forward model presented. A two-layer approach towards grasping has been presented: (i) position control through feedback, which is the traditional approach – but delays in the feedback cycle can generate instabilities; (ii) the addition of a feedforward predictive capability to partially circumvent this problem of instabilities by adopting pre-trained set of neural networks which in a way emulates the function of the cerebellum as seen in humans.

The predictive feedforward approach involves pre-learned models trained offline, which then provide a computationally efficient control model for low controller gains necessary for reactive and adaptive control. We have introduced task-specific models that are able to learn from their errors (make error predictions) under different and varying dynamics. The proposed approach is more practical for space-based manipulators because there would be no major hindrances such as high computational complexity; and secondly, the trained forward models do not require high computational resources to implement which is usually a constraint onboard spacecraft. This is where transfer learning comes in as a practical solution for transferring pre-trained earth-based model to space environment. Most automatic control algorithms have not been demonstrated in space as most manipulator control systems are teleoperated from earth.

Here, we present a forward model that is learned (or trained) as a neural network approximator using some trajectory datasets relating the output torque $\tau$ to the kinematic state of the joints $(\theta, \dot{\theta}, \ddot{\theta})^T$ in an experimental teaching mode (Fig. 2). The trained forward model will hence be able to take the analytically calculated torque (efference

copy of input motor commands) as its input, while the output of the neural network will be the predicted trajectory output $(\theta_{ff}, \dot{\theta}_{ff}, \ddot{\theta}_{ff})^T$. The system then incorporates an inverse model with a feedforward adaptive part; that is, it includes a feedback loop and feedforward component. The feedforward controller is trained using the output of the feedback controller which serve as error signals. The trained feedforward component models the inverse dynamics of the system. The feedback controller is effectively a computed torque controller while the feedforward controller employs a gradient descent to minimize the error.

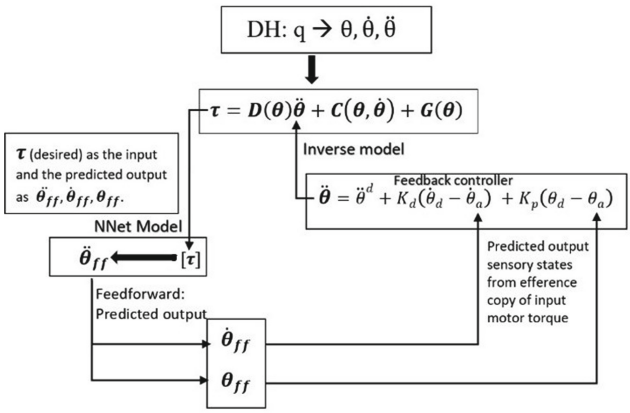


**Fig. 2.** The predictive forward model scheme. The neural network ("NNet") model is trained using data from experimental teaching mode. DH stands for Denavit–Hartenberg; q represents DH parameters for forward kinematics. D, C and G represent inertia matrix, coriolis and gravity components respectively; $K_d$ and $K_p$ are derivative and proportional controller gains.

The forward dynamic model of a robotic manipulator is given by (for the sensory joint acceleration rate):

$$\ddot{\theta} = D^{-1}(\theta)[\tau - C(\theta, \dot{\theta}) - G(\theta)]$$

Joint acceleration $\ddot{\theta}$ could be integrated to get joint rate $\dot{\theta}$ and joint rotation $\theta$ as the predicted sensory state outputs from torque input $\tau$. The body's muscular nature which produces a predicted trajectory output from efference input motor commands can be imitated by the predictive forward model [20]. To compensate for time delays, the feedforward control consequently predicts its response to system disturbances using a model of the plant process [21]. This predicted trajectory output would be supplied as input to the feedback component to compensate for delays (and this process could continue iteratively). It is believed that forward models can adjust 7.5 times more speedily than when using only inverse models [22]. The forward model, in this case, is executed as a neural network function estimator to the forward dynamics.

# 4 Methodology

We present here the configuration and kinematics of the space manipular adopted, and the mathematical implementation of the neural network multiple-target prediction algorithms.

## 4.1 Barrett WAM Configuration

For this study, the configuration of the WAM (Whole Arm Manipulator) representing the parameters of the manipulator at the initial (stowed) position are shown below as used in the simulation:

| $k$ | $a_k$ (m) | $\alpha_k$ (rad) | $d_k$ (m) | $\theta_k$ | Lower Limit ($\theta_{kL}$ rad) | Upper Limit ($\theta_{kU}$ rad) |
|---|---|---|---|---|---|---|
| 1 | 0 | $-\pi/2$ | 0 | $\theta_1$ | -2.6 | 2.6 |
| 2 | 0 | $\pi/2$ | 0 | $\theta_2$ | -2.0 | 2.0 |
| 3 | 0.045 | $-\pi/2$ | 0.55 | $\theta_3$ | -2.8 | 2.8 |
| 4 | -0.045 | $\pi/2$ | 0 | $\theta_4$ | -0.9 | 3.1 |
| 5 | 0 | $-\pi/2$ | 0.3 | $\theta_5$ | -4.76 | 1.24 |
| 6 | 0 | $\pi/2$ | 0 | $\theta_6$ | -1.6 | 1.6 |
| 7 | 0 | 0 | 0.06 | $\theta_7$ | -3.0 | 3.0 |

**Fig. 3.** D-H Table of the Barrett WAM.

The Barrett arm is a 7-DOF manipulator with a three-fingered hand as representative of an on-orbit servicing manipulator kinematics. For the kinematics solution, the procedure discussed and presented in Sect. 2 was implemented for the space manipulator. The procedure is the space-based kinematics shown for modifying terrestrial robots to space robots. The key to the space application approach is to replace the terrestrial parameters of $a_k$ and $d_k$ of Fig. 3 with the spaced-based equivalence [18], according to the lumped kinematic parameters as described in Sect. 2.

## 4.2 Implementation of Multiple-Target Prediction Algorithms

Machine learning algorithms were developed to learn/train the forward dynamics model by using the joint torques as input and the joint trajectories as targets. With respect to the nature of the trajectory datasets, and after various experimentations, the multiple-output regression tree and the multi-layer perception (MLP) multiple-output regression algorithms were identified for best prediction accuracy and computational efficiency. We present here the implementation of the algorithms, with emphasis on the multiple-output decision tree regression. Given a feature vector $x$, we aim to predict a vector of output responses $y$ using the function *h(x):*

$$\boldsymbol{x} = (x_1, x_2, x_3 \ldots, x_m) \overset{h(x)}{\Rightarrow} \boldsymbol{y} = (y_1, y_2, y_3 \ldots, y_d)$$

Some of the notable challenges are the proper modeling of dependencies among targets, i.e., between targets $y_1, y_2, y_3 \ldots, y_d$; and dealing with large number of multiple variable loss functions outlined over the output vector, $\mathcal{L}(\mathrm{y}, \mathrm{h}(\mathrm{x}))$.

There are two methods existing for multiple output regression, namely (i) problem transformation methods and (ii) algorithm adaptation methods. In our case, we used the algorithm adaptation method because it provides more accurate predictive performance, particularly in cases where there are correlations among the targets [23–25] - as we have in our robotic trajectory dataset, where the targets $(\theta, \dot{\theta}, \ddot{\theta})^T$ are co-related. Given a training dataset $D$ of $N$ samples containing a value assignment for individual variable $X_1, X_2, \ldots, X_m,\ Y_1, Y_2, \ldots, Y_d$; that is, $D = \{(\mathrm{x}^{(1)}, \mathrm{y}^{(1)}), \ldots, (\mathrm{x}^{(N)}, \mathrm{y}^{(N)})\}$. Every sample is categorized by an input vector of $m$ predictive variables $\boldsymbol{x}^{(l)} = (x_1^{(l)}, \ldots, x_j^{(l)}, \ldots, x_m^{(l)})$ and an output vector of $d$ target variables $\boldsymbol{y}^{(l)} = (y_1^{(l)}, \ldots, y_i^{(l)}, \ldots, y_d^{(l)})$, with $i \in \{1, \ldots, d\},\ j \in \{1, \ldots, m\},$ and $l \in \{1, ..., N\}$.

The aim is to learn multiple output regression model from $D$ comprising of a function $h$ that ascribe to each sample, given by the vector $\mathbf{x}$, and a vector $\mathbf{y}$ of $d$ target values:

$$h : \Omega_{X_1} \times ... \times \Omega_{X_m} \rightarrow \Omega_{Y_1} \times ... \times \Omega_{Y_d}$$
$$\mathrm{x} = (x_1, ..., x_m) \rightarrow \mathrm{y} = (y_1, ..., y_d),$$

where $\Omega_{X_j}$ and $\Omega_{Y_i}$ denote the sample spaces of each predictive variable $X_\mathrm{j}$, for all $j \in \{1, \ldots, m\}$, and each target variable $Y_\mathrm{i}$, for all $i \in \{1, \ldots, d\}$, respectively. The variables in targets $(\theta, \dot{\theta}, \ddot{\theta})^T$ are taken to be continuous, as it is the case for manipulators' joint trajectory. The trained multi-output model will be employed subsequently to concurrently predict the values $\{\hat{\mathrm{y}}^{(N+1)}, \ldots, \hat{\mathrm{y}}^{(N')}\}$ for all target variables of the new incoming unlabeled examples $\{\mathrm{x}^{(N+1)}, \ldots, \mathrm{x}^{(N')}\}$. Multi-target regression trees, as adopted in our case, can predict multiple continuous targets simultaneously. They have major benefits over adopting single regression tree for individual target [26]. Here, we have adopted an extension of the univariate recursive partitioning method (CART) [27] to our multi-target regression task. Therefore, the multivariate regression trees are modeled similarly to the steps followed in CART. The approach that performed better between MLP neural network and multi-target regression was chosen in each instance.

## 5 Results

Presented here are the results of the predictive feedforward model of the Barrett WAM space manipulator. The significance of the result is to demonstrate how we have developed neural network and regression models which are capable of predicting (to a high degree of accuracy) forward trajectory variables $(\theta_{ff}, \dot{\theta}_{ff}, \ddot{\theta}_{ff})$ from an efference copy of the torque, as shown in Fig. 4. It means the models are poised to cancel the sensory effects of the arm movement, providing anticipated sensory consequences from the motor command. With this, instabilities that could arise in delays when using traditional feedback cycle have been partially circumvented. This is akin to how the human cerebellum functions as discussed in Sect. 3. Here, we adopt a dataset publicly made available by [28], where the Barrett WAM was taken by the end-effector and guided along several trajectories in a teaching mode – in this case, it implies that sensor noise was included in the training dataset. During the imagined motion, the joint trajectories $(\theta, \dot{\theta}, \ddot{\theta})$ were sampled from the robot and the corresponding motor torques $(\tau)$ measured for each data point. The dataset has a total of 12,000 samples. For the 7 degree-of-freedom (7-DOF)

Barrett arm, 7 motor torques were measured, along with 21 joint trajectory variables representing seven joint angles ($\theta$), seven joint velocities ($\dot{\theta}$) and seven joint accelerations ($\ddot{\theta}$).

Deep learning neural network models and multiple-output regression models were developed, and we learned the forward dynamics model by using the joint torques as input and the joint trajectories as targets. Given a feature vector x, we aim to predict a vector of output responses y using the function h(x):

$$\boldsymbol{x} = (x_1, x_2, x_3 \ldots, x_m) \overset{h(x)}{\Rightarrow} \boldsymbol{y} = (y_1, y_2, y_3 \ldots, y_d)$$

**Table 1.** Prediction accuracy of joint angles (Data point 243).

| Joint number | Joint angle (rad) *Test Set* | Joint angle (rad) *Predicted* | Accuracy (%) |
|---|---|---|---|
| 1 | 0.0814 | 0.0799 | 98.2 |
| 2 | 0.6165 | 0.5696 | 92.4 |
| 3 | 0.0236 | 0.0222 | 94.1 |
| 4 | 1.745 | 1.6647 | 95.4 |
| 5 | 0.2123 | 0.199 | 93.7 |
| 6 | 0.0781 | 0.0751 | 96.2 |
| 7 | 0.0869 | 0.0844 | 97.1 |

The independent features were used to train each set of targets grouped separately by the joint angles, joint velocities, and the joint accelerations. Meaning three different models with different hyperparameters were trained (learned), in an attempt to manage target dependencies. The first model relates the 7 joint torques as input to the 7 joint angles as targets; the second model was learned between the 7 joint torques as input and the 7 joint velocities as targets, while the third model learned the relationship between the 7 joint torques as input and multiple-target prediction of the 7 joint accelerations as the output. It should be noted that the joint trajectory datasets for the training and testing were randomly split for better model learning and performance. In Table 1, a comparison for a chosen data point (sample number 243) for the predicted feedforward joint angles are shown, while Table 3 represents the case for joint velocities. These tables show the results comparing the predicted feedforward joint trajectory (for angle and velocity) to some desired joint trajectory specified as test set from the dataset. Table 2 shows the comparison for the accuracy of the end-effector's position in the three-dimensional cartesian space. This relates the desired joint angles and the predicted joint angles for the seven degrees of freedom, as carried out with space manipulator forward kinematics.

**Table 2.** Prediction accuracy of end-effector's position (data point 243).

| End-effector Cartesian position | Desired position | Predicted (actual) position | Accuracy (%) |
|---|---|---|---|
| x-position | 0.0040 | 0.0037 | 92.5 |
| y-position | 0.0047 | 0.0046 | 97.9 |
| z-position | 0.9008 | 0.9005 | 99.9 |

**Table 3.** Prediction Accuracy of Joint Velocities (Data point 243).

| Joint number | Joint velocity (rad/s) Test Set | Joint velocity (rad/s) Predicted | Accuracy (%) |
|---|---|---|---|
| 1 | 0.0657 | 0.0647 | 98.5 |
| 2 | −0.1850 | −0.1835 | 99.2 |
| 3 | −0.1794 | −0.160 | 89.2 |
| 4 | −0.0678 | −0.0646 | 95.3 |
| 5 | −0.0235 | −0.0231 | 98.1 |
| 6 | 0.0478 | 0.0434 | 90.8 |
| 7 | −0.0895 | −0.0872 | 97.4 |

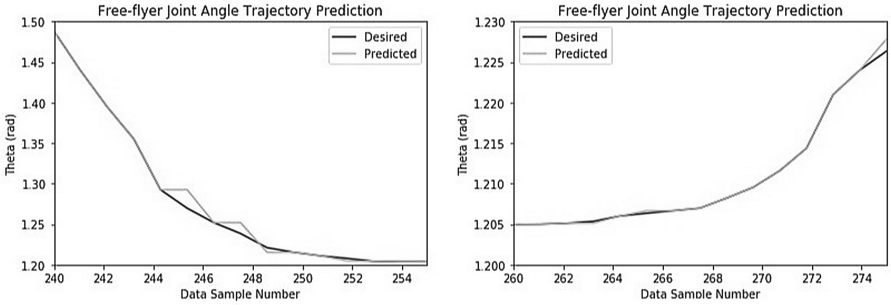


**Fig. 4.** Predicted-to-desired, data points 240–255; 260–275, for joint angle 1, simulated free-flyer.

Figure 4 represent results of several sample data points for the simulated Barrett WAM free-flyer, showing prediction correlation along the path of joint-angle motion ($\theta$) for a single chosen joint (Joint 1). The joint trajectory dataset from the terrestrial case was simulated and modified following the kinematics and dynamics of robotic free-flyer presented in Sect. 2. Initially, the free-flyer dataset was tested on the predictive model built earlier for the terrestrial case. The performance was, surprisingly, poor and unsatisfactory. However, retraining the earth-based model using the newly gotten free-flyer space dataset resulted in highly correlated trajectory accuracy between the desired

and predicted as shown in Fig. 4 – without the need to tune the hyperparameters. The drawback to the feedforward model here is the need to re-learn the new dataset as adopted for the space manipulator. It was discovered that the possibility of transfer learning could not be exploited after the optimized initial training, even though the deduced terrestrial-to-space manipulator dynamics were incorporated analytically in the model learning process. A second limitation was the fact that a change in dynamics (such as time delays and/or payload variations) resulted in the necessity to retrain the model, and learned weights were not transferable. Again, the controlling equations and dynamics of the terrestrial-to-space manipulator have not changed, these limitations should have been learned (accounted for) in the trained model as part of the space robot's dynamics.

The prediction accuracy in both cases ranged between 89–99%, and our model compared favorably to the expected feedforward joint trajectory. In the simulation carried out, low gains were required subsequently in the feedback controller for trajectory tracking. The feedforward model provided good anticipatory sensory consequences, although a morphing system will be required to transform between the terrestrial and free-flyer dynamics. For the variations seen in the prediction accuracies, this issue will apply to any serial manipulator control system.

## 6 Conclusion and Future Work

We have introduced forward models implemented as trained neural networks as a means of supplementing traditional feedback controllers for space manipulators. The forward models show high accuracy predictions for the feedforward joint trajectory; robust enough to provide a platform for reactive manipulation (because of low gains in the feedback controller), in a way to circumvent the sensor-dependent traditional approach. Different forward model trainings were required for the terrestrial and space robot's joint trajectory predictions for high accuracy, although hyperparameter tuning was not required, rather a case of model re-training on new datasets (dynamics and environment). Therefore, there might be a requirement for some offline adaptation or implementation of morphing approach between the terrestrial and spaced-based dynamics. This exposes severe problems in neural networks including deep learning as a model of general intelligence – transfer learning lacks the adaptability and requires a profusion of motor models, and the introduction of payloads and force control exacerbates this. Yet, the human cerebellum can readily adjust to the space environment within seconds and does not have to start learning from motor babbling as implied by the neural network model. It is clear as detailed in Sect. 2 that the terrestrial kinematics and the space kinematics are of the same form but with only changes in parameters. Given that the two equations are similar, the two polynomial curve shapes should be similar but shifted in multidimensional space to match their specific input-output mappings. Transfer learning cannot seem to shift one polynomial curve fit onto the other, but the human cerebellum can. The problem is that input-output mapping that a neural network learns does not retain the kinematic/dynamic structure. This is a practical limitation of regression and neural network models as shown in this work.

An obvious approach to provide a solution would be to explore reinforcement learning algorithm. An adaptive feedforward model is pre-trained on earth prior to deployment and subsequently adapts to changes in the dynamics and other environmental parameters of the system in space. A recent suggestion to such an approach was covered in Rapid Motor Adaptation algorithm [29]. This is an approach that we are currently exploring its feasibility for adaptive and compliant space manipulator control transferable from learned model. Continuous online training as typically obtained in the context of traditional reinforcement learning deployment is not feasible in the application of space robotics, for reasons stated in Sect. 3 – constraint of high computational resources onboard spacecraft.

## References


1. Bullock, D., Grossberg, S.: Cortical networks for control of voluntary arm movements under variable force conditions. Cereb. Cortex **8**(1), 48–62 (1998)
2. Kawato, M., Furukawa, K., Suzuki, R.: Hierarchical neural network model for control and learning of voluntary movement. Biol. Cybern. **57**, 169–185 (1987)
3. Ellery, A.: Tutorial review of bio-inspired approaches to robotic manipulation for space debris salvage. Biomimetics J. **12**(5), E19 (2020)
4. Shrivastava, S., Modi, V.: Satellite attitude dynamics and control in the presence of environmental torques—a brief survey. J. Guid. Control. Dyn. **6**, 461–471 (1983)
5. Vijayakumar, S., Schaal, S.: Locally weighted projection regression: Incremental real time learning in high dimensional space. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 1079–1086 (2000)
6. Nguyen-Tuong, D., Peters, J. R., Seeger, M.: Local Gaussian process regression for real time online model learning. In: Advances in Neural Information Processing Systems (NIPS), pp. 1193–1200 (2008)
7. Gijsberts, A., Metta, G.: Real-time model learning using incremental sparse spectrum Gaussian process regression. Neural Netw. **41**, 59–69 (2013)
8. Meier, F., Hennig, P., Schaal, S.: Incremental Local Gaussian Regression. In: Advances in Neural Information Processing Systems, pp. 972–980 (2014)
9. Jamone, L., Damas, B., Santos-Victor, J.: Incremental learning of context-dependent dynamic internal models for robot control. In: Proceedings of the IEEE International Symposium on Intelligent Control (ISIC), pp. 1336–1341 (2014). https://doi.org/10.1109/ISIC.2014.6967617
10. Toussaint, M., Vijayakumar, S.: Learning discontinuities with products-of-sigmoids for switching between local models. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 904–911 (2005)
11. Petkos, G., Toussaint, M., Vijayakumar, S.: Learning multiple models of non-linear dynamics for control under varying contexts. In: International Conference on Artificial Neural Networks, pp. 898–907. Springer (2006)
12. Wolpert, D.M., Kawato, M.: Multiple paired forward and inverse models for motor control. Neural Netw. **11**(7–8), 1317–1329 (1998)
13. Ross, J., Ellery, A.: Panoramic camera tracking on planetary rovers using feedforward control. Int. J. Adv. Rob. Syst. **4**, 1–9 (2017)
14. Lindberg, R., Longman, R., Zedd, M.: Kinematics and reaction moment compensation for the spaceborne elbow manipulator. In: 24th Aerospace Sciences Meeting, AIAA-86-0250, Nevada (1986)

15. Longman, R., Lindberg, R., Zedd, M.: Satellite-mounted robot manipulators—new kinematics and reaction compensation. Int. J. Robotics Res. **6**(3), 87–103 (1987)
16. Vafa, Z., Dubowsky, S.: On dynamics of manipulators in space using the virtual manipulator approach. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 579–585 (1987)
17. Vafa, Z., Dubowsky, S.: Kinematics and dynamics of space manipulators: the virtual manipulator approach. Int. J. Robotics Res. **9**(4), 852–872 (1990)
18. Ellery, A.: An Introduction to Space Robotics. Praxis–Springer Series on Astronomy and Space Sciences. Praxis Publishers (2000)
19. Tovée, M.J.: Neuronal processing: how fast is the speed of thought? J. Curr. Biol. **4**(12), 1125–1127 (1994)
20. Morasso, P., Baratto, L., Capra, R., Spada, G.: Internal models in the control of posture. Neural Netw. **12**, 1173–1180 (1999)
21. Basso, D., Belardinelli, O.: Role of the feedforward paradigm in cognitive psychology. Cogn. Process. **7**, 73–88 (2006)
22. Flanagan, J., Vetter, P., Johansson, R., Wolpert, D.: Prediction precedes control in motor learning. Curr. Biol. **13**, 146–150 (2003)
23. Kocev, D., Džeroski, S., White, M.D., Newell, G.R., Griffioen, P.: Using single- and multi-target regression trees and ensembles to model a compound index of vegetation Condition. Ecol. Model. **220**(8), 1159–1168 (2009)
24. Breiman, L., Friedman, J.H.: Predicting multivariate responses in multiple linear regression. J. Roy. Stat. Soc. B **59**(1), 3–54 (1997)
25. Simila, T., Tikka, J.: Input selection and shrinkage in multi-response linear regression. Comput. Stat. Data Anal. **52**(1), 406–422 (2007)
26. Izenman, A.J.: Reduced-rank regression for the multivariate linear model. J. Multivar. Anal. **5**, 248–264 (1975)
27. Breiman, L., Friedman, J.H., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
28. Nguyen-Tuong, D., Seeger, M., Peters, J.R.: Model learning with local Gaussian process regression. Adv. Robot. **23**, 2015–2034 (2009)
29. Kumar, A., Fu, Z., Pathak, D., Malik, J.: RMA: Rapid Motor Adaptation for Legged Robots. In: The Robotics: Science and Systems (2021). https://doi.org/10.48550/arXiv.2107.04034