EECE 798K - Project Milestone
Data-Driven Modeling of 5-Dof KUKA Robot Manipulator

Malak Slim and Batool Ibrahim

***Abstract*— Robotic systems often lack accurate models, especially for complex structures with uncertain dynamics. This poses challenges for robust controllers design. Data-driven techniques provide a promising solution for this challenge. In this work, we aim to model the inverse dynamics of the 5-DoF KUKA youBot robotic manipulator, needed for implementing inverse dynamics controller. For this purpose, we collected data by maneuvering the robot to perform multiple trajectories, and implemented four different modeling approaches. Fully Connected Neural Networks (FCNNs), Long Short Term Memory (LSTM) networks, and Sparse Identification of Non-linear Dynamical Systems (SINDy), which rely solely on collected data. In addition, we implemented a hybrid model, which takes advantage of the available physical knowledge of robotic systems to enhance the reliability of the estimated dynamics. The four models were then evaluated on a separate dataset. Results show that the hybrid model outperforms the others. The code and data used in this work can be found in our GitHub repository at EECE798K-Project.**

I. Introduction

Recently, robots are getting involved in humans' daily life, which requires them to interact with different types of environments and address a diversity of tasks. This highlights the need for efficient and robust model-based controllers, such as the computed-torque control. However, the robustness of these controllers highly relies on the reliability and accuracy of the model of the robotic system under control. Depending on the type of controller, a different model is needed. For example, for velocity-based position control, the forward kinematics model of the robot is required. On the other hand, for computed-torque control, which compensates for dynamics such as gravitational and inertial forces, the inverse dynamics model of the robot is needed. Traditionally, differential equations of motion are derived by conventional methods such as Newton-Euler and Euler-Lagrange methods. However, analytical models do not capture all aspects of robotic systems and there remain unmodeled dynamics due to varying parameters, unmodeled friction or soft materials [1].

To address these issues, some libraries and software tools were introduced to estimate the different components of the kinematic and dynamic models of a robot system based on its Unified Robot Description Format (URDF) file, that defines the geometry as well as the kinematic and dynamic parameters of the robot. These libraries include the Robotics Systems Toolbox [2] on MATLAB and the Kinematics Dynamics Library (KDL) [3] on C++. Additionally, some high-fidelity simulators, such as Simscape Multibody and Gazebo, were used to perform accurate simulations of robotic systems. However, these tools cannot cope with changes of the robot structure and its environment, and thus cannot account for varying conditions and uncertainties. Moreover, the aforementioned simulators offer control design tools, but these tools depend on a linearized dynamic model, which makes them unreliable for intricate tasks that require robust controllers [4].

Fig. 1: KUKA YouBot Robtic Arm [5]

In this context, data-driven modeling is an alternative suitable technique to model non-linear robotic systems based on data, offering the advantage of capturing complex non-linear relationships between input and output variables. This capability enhances the performance of model-based controllers by providing accurate non-linear plant models derived from data, enabling more precise control and adaptation to varying operating conditions.

The aim of this project is to develop a data-driven based model of the inverse dynamics of the 5-Dof KUKA YouBot robot arm, represented by the equation below, using machine learning techniques:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)$$

where q is the (5×1) vector of joints positions, (5×1) vector of joints velocities, $M(q)$ is the (5×5) positive-definite mass matrix, $C(q,\dot{q})$ is the (5×5) centrifugal and Coriolis coefficient matrix and $G(q)$ is the (5×1) vector of gravitational torques. KUKA YouBot is a 5-joints robot arm, with a gripper connected to its end-effector, shown in Fig. 1. This makes it a suitable platform for different applications in the robotic research, including gripping, grasping, pick and place, etc.

The remaining part of this report is structured as follows: Section II provides an overview of related works in the literature. Section III describes the collected data used in the project. Section IV outlines all the proposed models. Section

V presents a detailed analysis of the obtained results. Finally, Section VI offers conclusions and future directions.

II. Literature Review

Supervised learning is a branch of machine learning where the model is trained on labeled data, typically represented as input-output pairs. When dealing with discrete outputs, the task is commonly known as solving a classification problem, whose aim is to assign input data points to predefined classes or labels. On the other hand, when handling continuous output values, the task is referred to as solving a regression problem, aiming to approximate the relationship between input variables and a continuous target variable. Learning the mechanical model of a robotic system is similar to solving a regression problem, where training samples are obtained from measurements of the states of the robot over time. The goal is to estimate the behavior of the robotic system over time by understanding the relationship between its states and actions.

Regression techniques were early introduced in the literature and they fall into two main classes: a mixture of linear models, such as Locally Weighted Projection Regression (LWPR), Gaussian Mixture Regression (GMR), and Extended Classifier System for Function approximation (XCSF), or a weighted sum of basis functions, such as Radial Basis Function Networks (RBFNs), Gaussian Process Regression (GPR), and Support Vector Regression (SVR). A review paper [6] shows that these models internally represent a ”unified regression model”, which relies on a superposition of Gaussian basis functions. Due to their focus on capturing local variations in the data and adapting their models accordingly, these algorithms, known as local learners, were widely utilized in the early literature for effective robot model learning [7]. A recent study employing this approach was presented in [8]. The authors utilized GPR to approximate the unknown forces in the forward dynamics of a 7-DoF robot arm. They proposed incorporating available physics-based knowledge, such as the system's kinematics or inertia parameters, into the GP modeling framework as prior knowledge. This significantly enhanced the data efficiency and prediction accuracy of the robot arm's behavior compared to existing methods in the literature, while respecting the physical constraints.

Another approach to non-parametric regression, aimed at creating more interpretable and transparent systems, includes symbolic regression [9] and sparse identification [10]. While symbolic regression focuses on discovering mathematical equations that represent the underlying relationships in the data, sparse identification aims at finding concise models under only a limited number of key terms that dictate the dynamics, resulting in equations that are sparse in the space of potential functions. In fact, this assumption is valid for numerous physical systems. The developed algorithm is called Sparse Identification of Nonlinear Dynamics (SINDy). Parallel Implicit Sparse Identification of Nonlinear Dynamics (SINDy-PI) [11] extended the SINDy framework to identify implicit dynamics more robustly, particularly in the presence of noise. Inspired by SINDy-PI, a new approach was proposed in [4] to derive accurate dynamic equations of serial manipulators such as a two-link KUKA robot. The framework starts by building a library of mathematical functions which would capture the relationship between the system's input and output. Then, using a training dataset, SINDy-PI algorithm is used to obtain the best models from the library in addition to the appropriate coefficients which reflects the dynamic equation of the robot.

In contrast to non-parametric models, which learn the structure of the relationship between inputs and outputs directly from the data, without making assumptions about the specific form of the relationship, neural networks appear in the literature of regression problems as a form of parametric models. They are based on assuming a fixed structure of the model and learning a fixed set of parameters from the training data. These parameters include the weights and biases associated with the connections between neurons in the network. However, the power of neural netowrks lies in their ability of capturing high non-linearities in data. This makes them a good fit for modeling robotic systems. For instance, the authors in [12] used a single hidden layer feedforward neural network to model the forward kinematics of a UR5e robot and proposed an iterative learning algorithm to update its weights. They used the 6 joints angles as input data and the 3 coordinates of the robot end effector position in Cartesian space as output data. Similarly, in [13], the authors employed multilayer perceptron (MLP) neural networks to model the dynamics of an industrial robotic manipulator. They trained them using a randomized search for hyperparameter tuning and AI generated data set. Another work [1] proposed a deep neural network learning algorithm which consists of a self-organized and a recursive layer. They used two publicly available and three self-recorded datasets from four robots (Sarcos, Barrett, Baxter and UR10).

Recurrent Neural Networks (RNNs) have gained significant interest in data-driven modeling of robot dynamics. This is due to the sequential nature of the used and collected sensor data, making RNNs a good choice for handling such datasets. Among the most commonly used RNN architectures in this field are the Gated Recurrent Units (GRUs) and the Long Short-Term Memory (LSTM) networks. The researchers demonstrate that LSTM and GRU outperform classical regression-based techniques in terms of model learning performance. For instance, the study in [14] demonstrated the adaptability and efficiency of these RNN models in addressing real-time model variations and uncertainties, particularly in scenarios with higher DoF and dynamic environments. LSTM is used in [15] to model the inverse dynamics in 7-DoF hydraulic manipulator. The authors investigated the effect of hyperparameter settings on the LSTM model performance in such problem, in addition to the reduction of the inputs from the combination of joint position, velocity, and acceleration into only joint position while keeping acceptable performance for torque prediction.

While the models mentioned above rely solely on data-driven techniques, some researchers were interested in the integration of both analytical and data-driven models, arguing that this hybrid approach can yield superior models. An example is demonstrated in [16], where a hybrid model, combining an analytical model with a learned error model, was employed to model the inverse velocity kinematics of the Bionic Handling Assistant (BHA) and the inverse dynamics of the 7-joints rigid industrial manipulator KUKA Lightweight Robot (LWR). In both cases, the error model was trained using Extreme Learning Machines (ELMs), which are feed-forward neural networks with a single hidden layer and an efficient training scheme based on linear regression. Similarly in [8], the authors leverage the strengths of GPR as well as the physical knowledge of the system to model a robot arm's forward dynamics.

III. Data Description

In this work, we aim to model a real robot using data-driven techniques and validate it through real-life experiments. Therefore, we focused on gathering realistic measurements directly from the sensors onboard the robot. By doing so, we anticipate capturing the most reliable dynamics of the robot manipulator, thus ensuring the robustness and applicability of our framework in practical settings.

A. Sources

The 5-Dof KUKA YouBot has an encoder and a current sensor on each joint. It also offers a C++ API, called YouBot driver [17], which is a set of C++ classes and functions that allows reading measurements from the sensors onboard KUKA in real-time. It provides position measurements (q) directly from encoders, velocity measurements ($\dot{q}$) derived from position measurements, and torque values (τ) calculated from current measurements using the motor constant of each actuator. To get the joints accelerations ($\ddot{q}$), we can then derive velocity measurements with respect to time. With the youBot driver, our approach involves utilizing C++ code to save recorded data into CSV files while running our experiments on KUKA.

B. Data Collection

To collect data, twelve experiments were conducted, each involved maneuvering the KUKA robot along a prefined distinct trajectory. For each trajectory, we recorded 15 measurements at a sampling frequency of $1000Hz$. These include the position, velocity, and torque measurements, for each of the 5 joints. The number of data samples collected from the 12 experiments with all trajectories is 8513 samples. Each data sample corresponds to the input-output pairs $(q_t, \dot{q}_t; \tau_t)$ at a specific time instant t. The collected data was then split into training, testing, and validation sets by selecting data samples corresponding to 8 random trajectories for training and 2 random trajectories for monitoring the training process on validation data. The trained model was then tested on the data samples corresponding to the remaining 2 trajectories.

The trajectories followed by the robot arm end-effector in the 12 conducted experiments are represented in Fig. 6.

C. Data Pre-processing

To facilitate the preparation of data for training and testing the machine learning models, we used the Pandas library from Python to read the recorded measurements from 12 CSV files, each corresponding to one of the 12 trajectories, into data frames. Subsequently, we concatenated all measurements into input and output matrices, containing the values of joints positions and velocities, and the values of joints torques, at each time step for all trajectories, respectively. The structure of the prepared input and output matrices is depicted in Fig. 2.

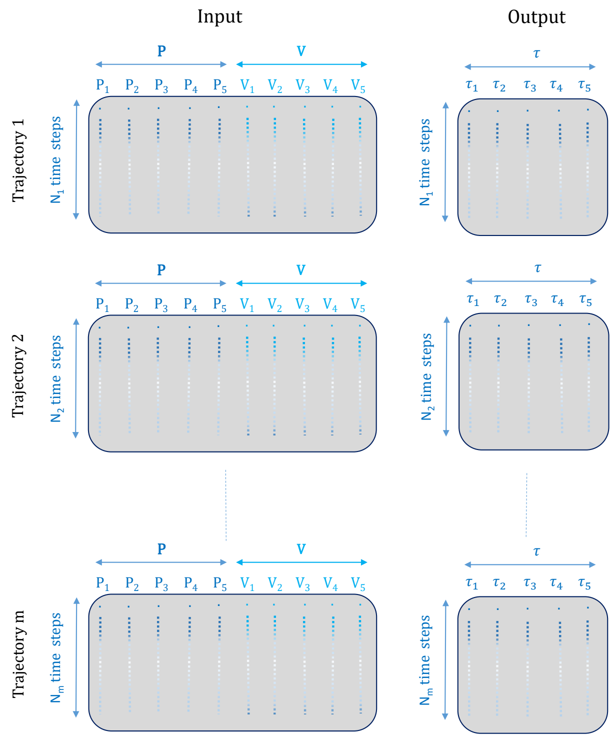


Fig. 2: Final structure of the pre-processed data.

IV. Methodology

After collecting and pre-processing the data, we began experimenting with various machine learning approaches to construct a model that represents the inverse dynamics of the KUKA youBot. Such a model should take as inputs the position and velocity of each joint, and outputs the corresponding joint torque (see Fig. 3).

Remark 1: In this work, we opt to use solely position and velocity measurements as input data. This decision is supported by the findings in [16], which demonstrate that similar performance can be achieved by using either (q, $\dot{q}$, $\ddot{q}$; τ) or (q, $\dot{q}$; τ) as input-output pairs for learning the inverse dynamics of a robotic system. By doing so, we avoid the derivation of velocity measurements, which tend to be noisy and might introduce distortions into the data.

As a first step, we attempted to use deep neural networks, which are expected to capture the non-linearity depicted in our data. Thus, we started with a Fully Connected Neural

Network (FCNN) and a Long Short Term Memory (LSTM). Then we switched to use Sparse Identification of Non-linear Dynamical Systems (SINDy) and a hybrid approach that combines data-driven techniques with available physical knowledge of robotic systems.

The approaches we have implmented and their corresponding architectures are detailed in the following subsections.

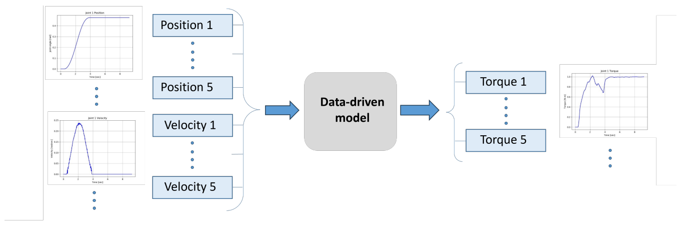


Fig. 3: Proposed Framework for learning inverse dynamics of the KUKA youBot.

A. *Fully-connected Neural Network (FCNN)*

First, we started by training a simple network with ten neurons at the input layer, five neurons at the output layer, and one hidden layer. The shapes of the input and output data used for training are the same as in Fig. 2.

By using early stopping and monitoring the validation loss, we began incorporating additional layers and adjusting the corresponding number of neurons and other hyper-parameters, such as the batch size and the number of epochs, until we converged to the deep network architecture shown in Fig. 4. The network was trained for 50 epochs with a batch size of 16 samples, using ReLU activation function in the hidden layers and a linear activation function in the output layer. We achieved a Mean Squared Error (MSE) of 0.168 on the training set and 0.562 on the validation set.

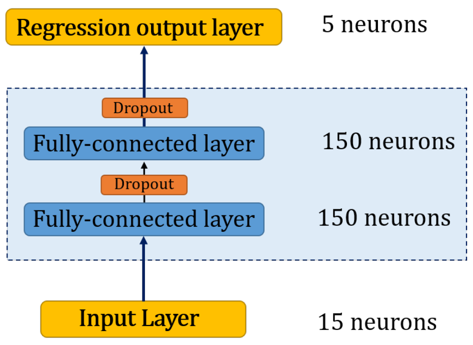


Fig. 4: Proposed FCNN architecture.

B. *Long-Short-Term-Memory (LSTM) Neural Network*

The advantage of LSTM networks in learning uncertainties in the manipulator dynamics was shown in [15], due to their ability in capturing sequential data and modeling long-term dependencies. Therefore, due to these advantages and the fact that our data is composed of time series, we attempted to utilize LSTMs for our modeling task.

1) Data preparation: For training the LSTM network, sequences of n samples are constructed. Each sequence is of the form: $\{(q_1, \dot{q}_1; \tau_1), (q_2, \dot{q}_2; \tau_2), ..., (q_n, \dot{q}_n; \tau_n)\}$. The number of samples per sequence was chosen to be 500. Hence, each trajectory was divided into a certain number of sequences, each of the same length (n=500). Remaining data samples of each trajectory were removed from the dataset. By doing so, we made sure that each sequence contains data samples from one type of trajectory only. This is to maintain the time-series property of the data, as it is a main characteristic of LSTM networks.

To increase the size of the training dataset, we selected the sequences from the 8 random trajectories chosen for training, by moving a fixed number of steps $s = 50$ in time. This means the the first sequence starts with the first data sample and spans the subsequent 500 samples, the second sequence starts from the 51^{th} data sample and includes the next 500 samples, and the process continues in a similar fashion for the remaining sequences. The resulting number of sequences is 117 for training, 2 for validation, and 3 for testing. The choice of $s = 50$ strikes a balance between augmenting the training dataset and mitigating redundancy in the data.

2) Model Architecture: To build our model, we started with a single LSTM layer, and similar to the case of FCNN, we fine-tuned the hyper-parameters using the validation data until we converged to the architecture presented in Fig. 5. Using early stopping to train the network for 200 epochs on a batch size of 64 samples, we achieved an MSE of 0.393 on the training set and 0.731 on the validation set.

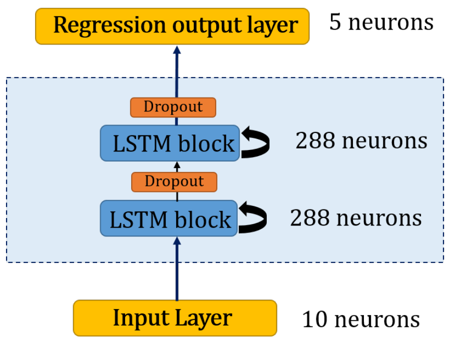


Fig. 5: Proposed LSTM architecture.

C. *Sparse Identification of non-linear Dynamics (SINDy)*

Sparse identification of nonlinear dynamics (SINDy) [10] has recently become a powerful tool for modeling complex non-linear systems. The motivation behind using SINDy for modeling the inverse dynamics of the 5-Dof KUKA youBot arises from its ability to provide an interpretable model represented in the form of differential equations. It achieves this by combining non-linear basis functions into a linear combination. Through solving a sparse regression problem, it determines the smallest number of terms necessary to accurately capture the underlying dynamics depicted in the data.

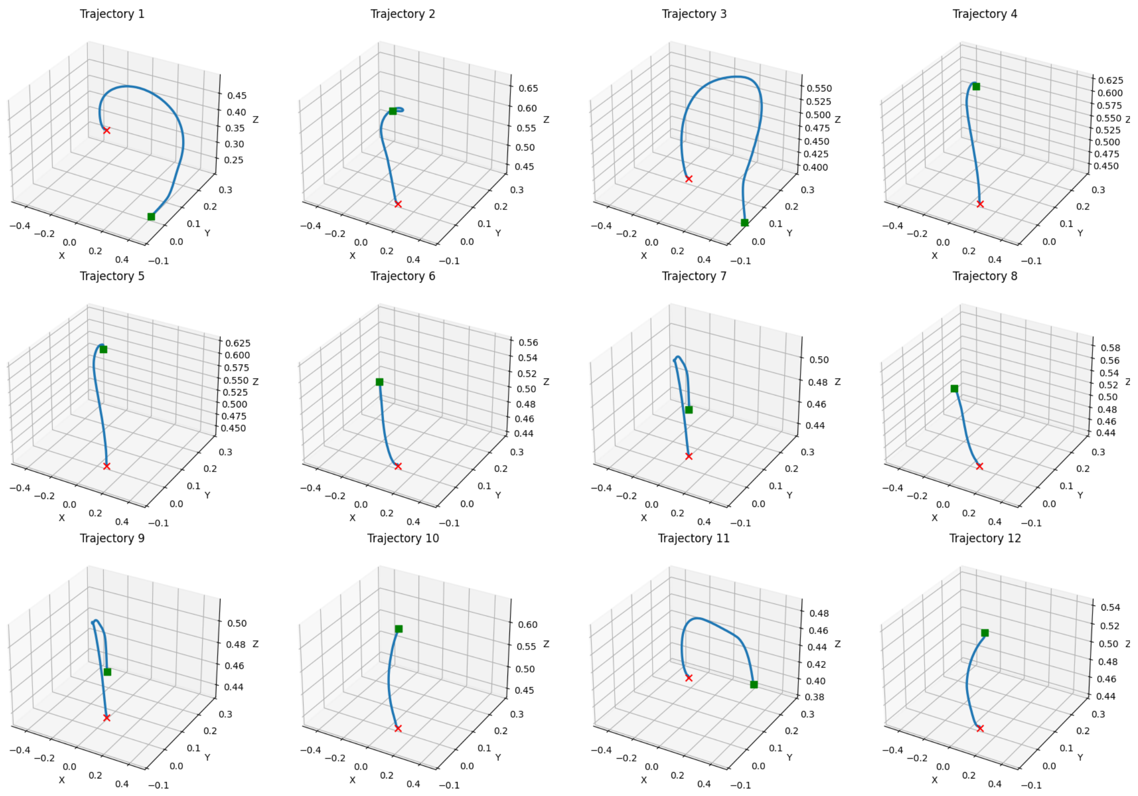


Fig. 6: Trajectories followed by the end-effector in the twelve experiments conducted during data collection. The markers in red and green indicate the start and end positions of each trajectory, respectively.

To define and train a SINDy model, we used "PySINDy" [18] library from Python. The primary step involves defining the feature library, which constitutes the set of non-linear candidate functions that are potentially present in the resulting inverse dynamics model. To properly define these terms, we rely on previous knowledge of simpler systems such as robotic manipulators with 2-Dof. These systems' inverse dynamics are typically accessible and derived through analytical methods like Newton-Lagrange, owing to their simple structure. Conversely, a robot with more complex geometry and 5-DoF introduces additional terms into the governing dynamic equation. These terms commonly comprise constants, trigonometric functions, and polynomials as elementary functions.

To accomplish this, we constructed a custom library including joints positions, joints accelerations, trigonometric functions applied to all possible combinations of joints positions summations, and second degree polynomials of joints velocities. Additionally, we specified possible combinations of multiple libraries together.

Next, we defined a SINDy model, that uses the constructed custom library as its feature set. We then fit the model into the training dataset, composed of 8 trajectories (as described in section III-B). We note here that in order to develop a comprehensive model through SINDy, it was imperative to incorporate joint accelerations within the training dataset. This inclusion was crucial for obtaining equations that encompassed terms related to joint accelerations. Consequently, we computed the joint accelerations by utilizing the gradient function available in NumPy. Training was done using Least Absolute Shrinkage and Selection Operator (LASSO) regression, which explores the feature space to identify the terms that most accurately represent the underlying dynamics of the data, while also promoting sparsity in the resulting model. As in the previous approaches, we tuned the hyperparameter of this model, namely the regularization parameter of the optimizer, α, on the validation dataset, to achieve the best possible performance while avoiding overfitting. This was achieved with a value of $\alpha = 0.029$, resulting with a MSE of 0.487 on the training set and 0.648 on the validation set. The resulting inverse dynamics model is represented by a set of five equations, relating the torque at each joint to the joints positions, velocities, and accelerations, as shown in table I.

D. Hybrid Inverse Dynamics

All previous models were purely data-driven, which estimate the inverse dynamics of the KUKA youbot solely based on the available collected data. However, relying only on data may not capture underlying physical dynamics, especially for highly nonlinear complex systems. While expanding the dataset and its diversity enhances the model's representation, it can be costly and sometimes unfeasible.

Hybrid models combine the available physical knowledge of a system with data-driven approaches. This has been

Joint	Differential Equation	
1	$-0.152\ddot{q}_1+0.019\cos(q_1)+0.323\dot{q}_1^2+0.019\cos(q_1)-0.423\cos(q_1+q_4)-0.250\cos(q_1+q_5)$ $-0.094\cos(q_1+q_2+q_4)+0.372\sin(q_1+q_3+q_4)+0.429\sin(q_1+q_2+q_4+q_5)$ $+0.266\ddot{q}_1\cos(q_5)+0.010\ddot{q}_2\cos(q_1)-0.047\ddot{q}_3\cos(q_1)$ $+0.030\ddot{q}_3\cos(q_2)+0.003\ddot{q}_4\cos(q_1)+0.129\ddot{q}_3\sin(q_1)$ $-0.025\ddot{q}_3\sin(q_2)+0.023\ddot{q}_4\sin(q_3)+0.964$	(1)
2	$-0.031\ddot{q}_1+0.053\ddot{q}_5-0.153\cos(q_1)+0.544\cos(q_2)+1.021\dot{q}_1-0.181\dot{q}_3-0.153\cos(q_1)$ $+0.544\cos(q_2)+-0.839\cos(q_1+q_4)+-1.261\sin(q_1+q_5)+-2.511\sin(q_2+q_3)$ $+0.927\sin(q_2+q_4)+0.067\cos(q_2+q_4)+1.412\cos(q_3+q_5)+-3.045\sin(q_2+q_3+q_4)$ $+0.098\sin(q_2+q_4+q_5)+0.229\ddot{q}_2\cos(q_2)+-0.468\ddot{q}_3\cos(q_2)+0.380\ddot{q}_3\cos(q_4)$ $+0.021\ddot{q}_4\cos(q_1)+0.171\ddot{q}_4\cos(q_2)+-0.271\ddot{q}_4\cos(q_3)+-0.014\ddot{q}_4\cos(q_5)$ $+0.057\ddot{q}_5\cos(q_5)+0.149\ddot{q}_1\sin(q_1)+-0.044\ddot{q}_1\sin(q_4)+-0.142\ddot{q}_2\sin(q_2)$ $+0.561\ddot{q}_3\sin(q_2)+-1.072$	(2)
3	$0.693\cos(q_2)-0.201\dot{q}_1+1.113\dot{q}_3+0.693\cos(q_2)-1.765\sin(q_2+q_3)-0.100\cos(q_2+q_4)$ $-0.242\sin(q_1+q_2+q_3)+0.687\cos(q_1+q_2+q_3)+0.368\cos(q_1+q_2+q_4)$ $-0.480\sin(q_2+q_3+q_4)-0.034\cos(q_1+q_2+q_3+q_5)-0.030\sin(q_1+q_2+q_4+q_5)$ $-0.056\ddot{q}_1\cos(q_1)+0.046\ddot{q}_1\cos(q_3)-0.008\ddot{q}_2\cos(q_2)-0.033\ddot{q}_3\cos(q_2)$ $+0.124\ddot{q}_3\cos(q_5)+0.048\ddot{q}_4\cos(q_1)-0.074\ddot{q}_4\cos(q_2)-0.059\ddot{q}_5\cos(q_5)$ $-0.006\ddot{q}_2\sin(q_2)-0.047\ddot{q}_2\sin(q_3)+0.130\ddot{q}_3\sin(q_3)+0.018\ddot{q}_4\sin(q_3)$ $+0.006\ddot{q}_5\sin(q_5)-3.621$	(3)
4	$-0.047\cos(q_2+q_4)+0.139\sin(q_1+q_4+q_5)-1.130\sin(q_2+q_3+q_4)+0.031\ddot{q}_2\cos(q_1)$ $-0.003\ddot{q}_3\cos(q_1)+0.016\ddot{q}_4\cos(q_3)-0.006\ddot{q}_1\sin(q_2)+0.026\ddot{q}_1\sin(q_3)$ $-0.080\ddot{q}_2\sin(q_2)+0.024\ddot{q}_3\sin(q_2)+0.012\ddot{q}_3\sin(q_3)+0.008\ddot{q}_4\sin(q_2)+0.216$	(4)
5	$-0.002\ddot{q}_1-0.044\cos(q_2+q_4)-0.107\cos(q_1+q_2+q_4)+0.009\ddot{q}_2\cos(q_2)$ $+0.009\ddot{q}_2\cos(q_3)+0.040\ddot{q}_5\cos(q_5)+0.023\ddot{q}_5\sin(q_5)-0.025$	(5)

TABLE I: Resulting inverse dynamics model of the KUKA youBot using SINDy.

proven to achieve more accurate dynamical models compared to pure analytical ones [16]. In the case of KUKA youBot, there exist analytical approaches, like the Euler-Lagrange method or the Newton-Euler method that are capable of calculating the analytical dynamics of our robot based on its properties. This is possible through KDL that uses Newton-Euler recursive algorithm to estimate the dynamics of a robotic manipulator given its URDF file, typically provided by the manufacturer. Hence, in our study, we established a hybrid model that integrates the available physical knowledge using KDL with a learned data-driven model.

To do so, for each collected trajectory in our data, we estimated the inverse dynamics (presented by the torque for each joint at each time step) analytically using KDL. The estimated joints torques are then fed as additional inputs to the hybrid model, along with joints positions and velocities, as illustrated in Fig. 7.

We used a FCNN as a backbone model for our hybrid approach. It takes 15 inputs instead of 10, considering the extra estimated torques $\{(q_1,q_2,q_3,q_4,q_5),(\dot{q}_1,\dot{q}_2,\dot{q}_3,\dot{q}_4,\dot{q}_5),(\hat{\tau}_1,\hat{\tau}_2,\hat{\tau}_3,\hat{\tau}_4,\hat{\tau}_5)\}$. Using the validation dataset to tune the hyper-parameters, we converged to the deep network presented in Fig. 8. We used Adam optimizer and early stopping to train the network for 50 epochs on a batch size of 64, using ReLU activation functions in the hidden layers and a linear activation in the output layer and achieved an MSE of 0.39 on the training set and 0.73 on the validation set.

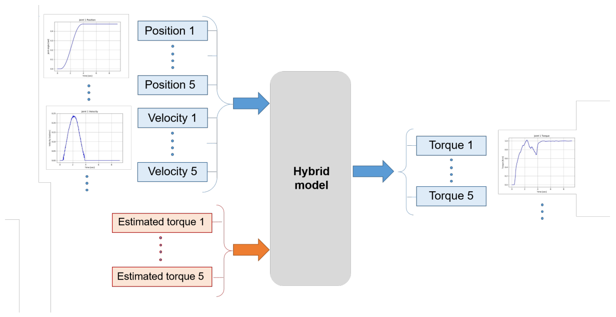


Fig. 7: Proposed hybrid model.

V. Results and Discussions

As a first attempt, we used our data exactly as it was recorded from the robot onboard sensors to build and test our four models. The corresponding MSEs on training,

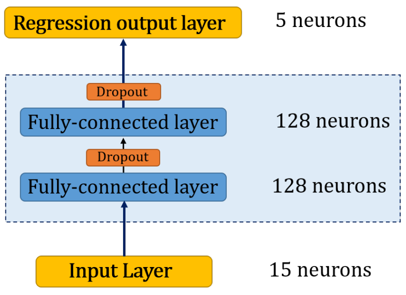


Fig. 8: Proposed FCNN architecture for the Hybrid model.

validation, and testing trajectories are presented in Fig. 9. SINDy's model performed the worse on all the trajectories by obtaining the highest MSE (> 2.1) on the testing data, highlighting its struggle in generalization. In contrast, the hybrid model demonstrated the best performance on both training and testing trajectories, achieving an MSE of less than 1.5 on the testing data.

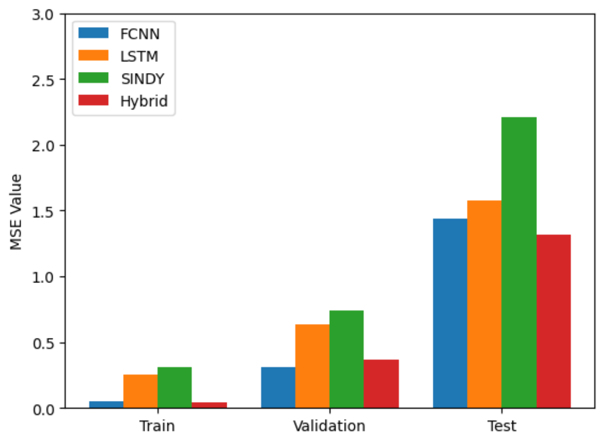


Fig. 9: MSE obtained before filtering the data across the four models

Due to the presence of noise in the collected sensory data, and in order to achieve better results, we filtered our data using a low-pass filter with a cut-off frequency of 10 Hz. An example of our filtering results is shown in Fig. 10, where the filter eliminates the high-frequency noise present in the original torque measurements, resulting in a smoother sequences.

We rebuilt and fine-tuned our models using the filtered data. The MSEs of our models after filtering are presented in Fig. 11. All the models performed similarly on the validation data. The hybrid model also achieved the best performance on both training and testing trajectories. This aligns with our motivation to use a hybrid approach. The integration of the physical knowledge of the KUKA youBot contributed in modeling more accurate dynamics compared to all the pure data-driven ones. Moreover, the hybrid model reached an approximate MSE of 0.9 on the test set, significantly lower than that observed before filtering, which highlights the importance of filtering the data before modeling. FCNN

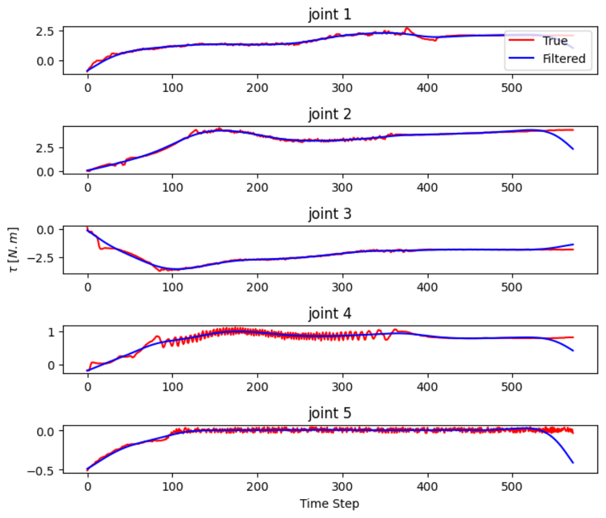


Fig. 10: Original and filtered torque data of the five joints on a sample trajectory.

performed the next best on the testing data with an MSE of 1.44, showcasing the ability of neural networks in capturing model complexities and non-linearities.

SINDy exhibited better generalization capabilities after filtering, with the MSE decreasing to approximately 1.5 on the test set. We noticed that SINDy removed some terms from the obtained equations after filtering, suggesting that it had initially added some complexity to model all the fluctuations present in the data before filtering.

Finally, LSTM performed worse on the testing trajectories with an MSE of approximately 1.9. This is contrary to our expectation given that the data used to build our model is made of time series. However, it may unfair to compare LSTM to other models solely based on MSE, because as mentioned previously, we are using 500 samples (as discussed in Section IV-B.1) for training and testing the recurrent network.

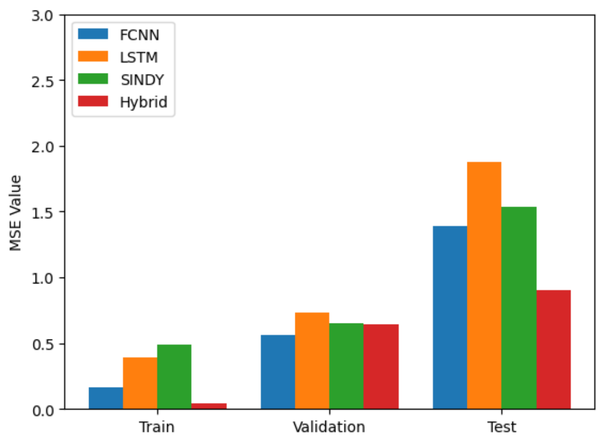
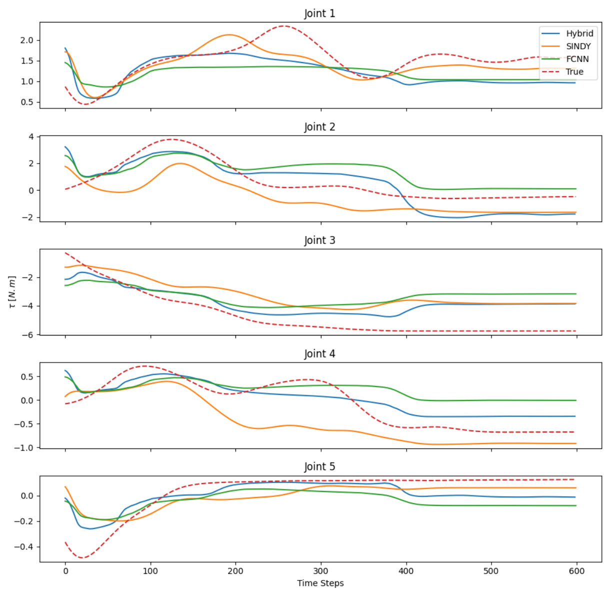
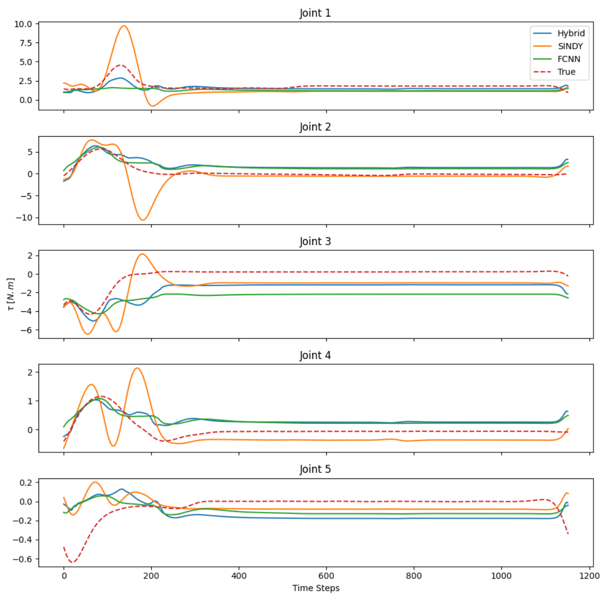


Fig. 11: MSE obtained after filtering the data across the four models

To observe the models' performances, we plotted the

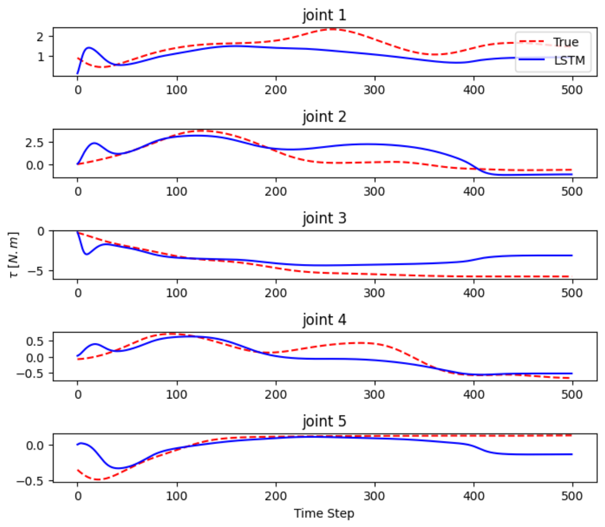


(a) First test trajectory

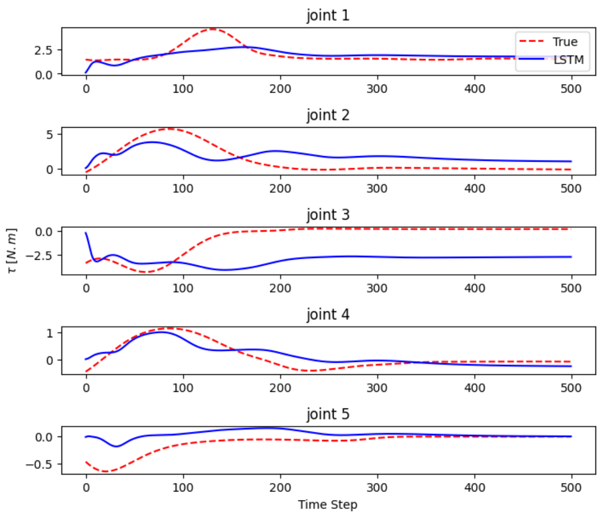


(b) Second test trajectory

Fig. 12: Predicted and true torque of the four models across the five joints on the test trajectories.



(a) First test trajectory



(b) Second test trajectory

Fig. 13: LSTM predicted and true torque of the five joints on the test trajectories.

predicted torques of the five joints on the test trajectories across the four models. Note that for the hybrid, FCNN, and SINDy models, we were able to plot the entire data samples (at all time steps) for each trajectory. The predicted torques for the two testing trajectories using the hybrid, FCNN, and SINDy models are presented in Fig. 12a and Fig. 12b alongside the actual torques. However, for LSTM, we used only 500 samples per trajectory and thus plotted the LSTM predicted torques independently in Fig. 13a and Fig. 13b for better plot presentation.

As shown in the figures, all models successfully generated the desired torques, however, there were varying percentages of error among the models, as previously discussed. All the models appear to perform worse on certain joints than others. For instance, the estimated torques for joint three, as depicted in Fig. 13a and Fig. 12a, exhibit poorer performance compared to other joints. This may arise from the complexity of this joint, as it is influenced by the other two joints, and hence posses intricate dynamics that are harder to model than other joints. Another reason could be the limited number of the collected trajectories, which may not encompass full diversity across this joint. One solution could be to collect more data that incorporates more diversity, thus enabling the models to capture the complexity and non-linearity across all the joints.

VI. Conclusion

In this project, we modeled the dynamics of the 5-DoF KUKA youBot robotic manipulator. Modeling the inverse dynamics of such robot is important to ensure the robustness of any corresponding model-based controller. We explored four different models: three purely data-driven models and one hybrid model that combines the advantages of the available physical knowledge of the robot with data-driven approaches. Evaluating the models on the testing trajectories has demonstrated the superiority of the proposed hybrid model over the other models. In future work, we aim to gather additional data that comprehensively capture the diversity of the model across all the joints, aiming for more precise dynamics.

References

[1] A. Polydoros, L. Nalpantidis, and V. Krueger, *Real-time Deep Learning of Robotic Manipulator Inverse Dynamics*, Sep. 2015.

[2] "Robotics System Toolbox." [Online]. Available: https://www.mathworks.com/products/robotics.html

[3] "orocos/orocos_kinematics_dynamics," Feb. 2024, original-date: 2013-10-16T20:10:49Z. [Online]. Available: https://github.com/orocos/orocos_kinematics_dynamics

[4] M. Omar, R. Li, and A. Asker, "A framework for data driven dynamic modeling of serial manipulators," *IEEE Access*, vol. 10, pp. 124 874–124 883, 2022.

[5] "youBot Store." [Online]. Available: http://www.youbot-store.com/developers/projects

[6] F. Stulp and O. Sigaud, "Many regression algorithms, one unified model - A review," *Neural Networks*, vol. 69, pp. 60–79, 2015, publisher: Elsevier. [Online]. Available: https://hal.science/hal-01162281

[7] O. Sigaud, C. Salaün, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: A survey," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129, Dec. 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188901100128X

[8] L. Rath, A. R. Geist, and S. Trimpe, "Using physics knowledge for learning rigid-body forward dynamics with gaussian process force priors," in *Conference on robot learning*. PMLR, 2022, pp. 101–111.

[9] C. Hillar and F. Sommer, "Comment on the article "Distilling free-form natural laws from experimental data"," Oct. 2012, arXiv:1210.7273 [physics]. [Online]. Available: http://arxiv.org/abs/1210.7273

[10] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data: Sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, Apr. 2016, arXiv:1509.03580 [math]. [Online]. Available: http://arxiv.org/abs/1509.03580

[11] K. Kaheman, J. N. Kutz, and S. L. Brunton, "Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics," *Proceedings of the Royal Society A*, vol. 476, no. 2242, p. 20200279, 2020.

[12] H.-T. Nguyen and C. C. Cheah, "Data-Driven Neural Network-Based Learning For Regression Problems In Robotics," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. Singapore, Singapore: IEEE, Oct. 2020, pp. 581–586. [Online]. Available: https://ieeexplore.ieee.org/document/9254442/

[13] S. Baressi Šegota, N. Anelić, M. Šercer, and H. Meštrić, "Dynamics modeling of industrial robotic manipulators: A machine learning approach based on synthetic data," *Mathematics*, vol. 10, no. 7, p. 1174, 2022.

[14] R. Mukhopadhyay, R. Chaki, A. Sutradhar, and P. Chattopadhyay, "Model learning for robotic manipulators using recurrent neural networks," in *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE, 2019, pp. 2251–2256.

[15] N. Liu, L. Li, B. Hao, L. Yang, T. Hu, T. Xue, and S. Wang, "Modeling and simulation of robot inverse dynamics using lstm-based deep learning algorithm for smart cities and factories," *IEEE Access*, vol. 7, pp. 173 989–173 998, 2019.

[16] R. Reinhart, Z. Shareef, and J. Steil, "Hybrid Analytical and Data-Driven Modeling for Feed-Forward Robot Control †," *Sensors*, vol. 17, no. 2, p. 311, Feb. 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/2/311

[17] youbot, "youbot/youbot_driver," Jan. 2024, original-date: 2011-01-24T16:14:54Z. [Online]. Available: https://github.com/youbot/youbot_driver

[18] "PySINDY." [Online]. Available: https://pysindy.readthedocs.io/en/latest/