Fakultät für Informatik
Lehrstuhl für Echtzeitsysteme und Robotik

Supervised Learning for Robot Workspace Identification

Lukas Hornik, Xinyu Chen, Theresa Bruns

Practical Course *Reinforcement and Feature Learning for Robotic Manipulators*, Summer 2023

Advisor:	Jonathan Külz
Supervisor:	Prof. Dr.-Ing. Matthias Althoff
Submission:	11. August 2023

Supervised Learning for Robot Workspace Identification

Lukas Hornik, Xinyu Chen, Theresa Bruns
Technische Universität München
Email: l.hornik@tum.de, Xinyu.Chen@in.tum.de, theresa.bruns@tum.de

***Abstract*—The problem of accurately and efficiently describing a robot's workspace is of high importance for a range of tasks like motion planning and control. This problem becomes more complicated in the field of modular robotics specifically, due to the flexibly changeable characteristics of the manipulator. With traditional methods being infeasible for these types of robots, we develop a novel, generic framework for workspace identification based on the supervised training of a neural network. We apply our method to the setting of robot movements restricted to a 2D plane and distinguish two prediction tasks for binary reachability and dexterity in the form of a manipulability index. Results demonstrate the effectiveness of our approach and indicate the generation of highly precise workspaces especially for modular robots with higher degrees of freedom.**

(*TB*) - August 11, 2023

I. Introduction

Workspace identification and generation is a crucial task in the field of robotics [1]. The manipulator workspace, which is defined as the set of all poses reachable by an end-effector with certain joint angles, provides useful information about the utility and capabilities of a robot [2], [3]. Determining and extracting details about the individual workspace of a robot is useful not only in terms of identifying reachable points, but also for knowing how well certain positions can be attained to fulfill the desired task [2].

Understanding and effectively handling the workspace is essential to all robotic tasks like motion planning and control, task design, grasp feasibility analysis and selection as well as robot mounting point or assembling strategy optimization [1], [3], [2]. Usually, for classical robot arms with a predefined structure, workspace identification is typically done with either forward or inverse kinematic methods [2]. These strategies are not suitable for the task at hand due to the fact that the types of robots that are subject of this proposal are Modular Reconfigurable Robots (MRR) [4].

MRRs are a recent, innovative advancement in the field of industrial robotics. They have been developed with a desire for higher flexibility and increased cost-efficiency in comparison with standard, readily assembled robotic manipulators. Being composed of a set of exchangeable modules, MRRs come with the advantage that they can be flexibly adapted to several different tasks and more easily maintained at the same time [4]. But the fact that these modular robots are reconfigurable renders the task of workspace identification more difficult in the sense that, with each re-assembly of the modules, a new, completely different resulting workspace is created. Thus, using conventional analytical methods like forward or inverse kinematics would come with an immense computational cost, which makes them infeasible.

This motivates the development of a more robot-generic approach for efficient workspace identification. Since the correlation between a robot's configuration and its reachable space is of a highly non-linear nature, adopting neural networks for this task promises to be a suitable strategy. Therefore, we present a novel method based on a supervised learning pipeline that can generalize the mapping from any modular robot to its respective reachable workspace.

We divide our approach in two different sub-tasks for generating workspace maps based on simple binary reachability on the one hand, and also including some notion of dexterity at a certain point in the form of a manipulability index on the other hand (see subsection IV-A1). While the former is useful for static, motionless tasks, the latter also provides information for dynamic situations, in which the end-effector requires some ease of motion or local flexibility [2], [3], [5].

In order to provide detailed insights into our approach, we start off by putting our method into the context of relevant existing works in the literature review section II. Then, in section III, an overview of our concept is presented before elaborating on the proposed method and implementation including both the data generation part and the learning pipeline itself in sections IV-A and IV-B. Based on that, section V delivers the final results as well as their evaluation. Finally, some future elements of work are identified with the conclusion in section VI.

(*TB, XC*)

II. Literature Review

The literature review is divided into a review of classical methods for robot workspace identification, a brief introduction of adjacent work with learning-based approaches, and a more general revision of modular reconfigurable robots.

(*XC, LH, TB*)

A. Classic Robot Workspace Identification

With discretized task space as a grid, the workspace of a robot can be initially interpreted by a reachability map, in which each cell is assigned a binary value to indicate if it can be reached or not. In this identification manner, forward kinematics and inverse kinematics are two underlying principles. Either or both are applied in the current state of

the art [1], [2], [3], [6]. In the case of forward kinematics, joint configurations are randomly sampled. Based on that, the corresponding 3D positions, or the 6D parameters in case orientations are considered as well, of the tool center point (TCP) are computed. This allows to generate numerous TCP samples in a relatively short time, but full coverage is not guaranteed. When inverse kinematics is implemented, all cells in the discretized task space are examined to see if any joint configuration is yielded, which ensures completeness. However, the time complexity increases drastically compared to the forward kinematics [2], [3].

Besides the binary reachability value, dexterity measurement can be included for more sophisticated insights into the workspace, for instance, manipulability index introduced by Yoshikawa et al. [7]. It is one of the most commonly applied metrics for dexterity representation, which describes the distance a robot has to a singularity at a certain pose. There are several variations to augment this technique so that the value can be sensitive to the joint limits and obstacle positions and irrelevant to the robot scale [8], [9]. In addition, the dexterity values can be obtained based on inverse kinematics according to Zaccharias et al. [3], who compute the percentage of reachable bins within a voxel of the grid world after uniformly sampling hundreds of bins in each one. For simplicity, the original Yoshikawa manipulability index is used in this work (see IV-A).

(*XC, TB*)

B. Learning-based Method on Joint Space Prediction

A related idea to workspace identification with learning-based methods is the prediction of the configuration space with learning-based approaches. Recently, Benka et al. [10] had remarkable results using a convolutional encoder-decoder framework to predict the joint space for 2D workspace. Their work inspires our general approach of predicting workspace with Long Short-term Memory (LSTM) [11] models, which is widely used for its eponymous advantage such that the correlation of the elements within a sequential sample can be encoded in a sensible way. This feature can help the model to gain reasonable understanding of the entire arrangement of sequential robots [12].

(*LH, XC*)

C. Module Reconfigurable Robots

Modular reconfigurable robots (MRRs) are a class of robots that are designed to be arranged from a set of primitive components in order to adapt to a specific task. In recent years, MRRs gained significant attention. Especially, their promise of high versatility combined with low build cost sparks interest beyond experimental research [4], [13], [14]. To simulate MRR in a standardized and efficient way, *Timor-python*, an open-source library for model generation and task-based configuration optimization, was recently introduced by Külz et al. [4]. This library is also used as the underlying software foundation of this project with regard to robot representation, generation, and display (described in more detail in IV-A). In the following section, the general project structure will be explained.

(*LH, XC*)

III. Concept Overview

As motivated in the introduction, we try to predict the workspace of modular reconfigurable robots using a neural network. The broad idea is to feed a model with encoded robots, which then in turn returns their associated workspaces. To achieve this, the task is broken down into smaller steps. First, robots and their workspaces (including manipulability) have to be generated. Additionally, suitable representations for these have to be identified. After this data generation step, the supervised learning phase can begin. This second step covers the two sub-tasks of binary workspace prediction and prediction of workspaces including manipulability.

Over the course of the project, we constrained the problem to robots only capable of moving in a 2D plane and prediction of workspace in terms of 3D (positions) instead of 6D (positions and orientations). We also limit the robots to be only of sequential structure. Therefore, not allowing any Y-shaped or circular robots. Further constraints on robot length, number of joints and modules are made for the data generation phase (see sections IV-A2 and IV-A3 for more details). A detailed explanation of our implementation is given in the next chapter.

(*LH, XC*)

IV. Proposed Method and Implementation

Following the described solution concept, this section further elaborates on the proposed implementation and provides details on both necessary constituents. First, subsection IV-A delves into the steps taken for the custom data generation process. Based on this, the supervised learning pipeline and its individual components are explained in subsection IV-B.

(*TB, LH, XC*)

A. Data Generation and Processing

For the models to be able to learn properly, it is necessary to generate high quality data of sufficient size. Therefore, we have to come up with efficient ways of robot and workspace generation. In section IV-A1 we identify a suitable representation for the workspaces and also discuss how to augment it using Yoshikawa's "Manipulability Index" [7] as introduced in chapter II-A. Next, the implementation and encoding of robots is specified in IV-A2. After which we explain the data generation pipeline in IV-A3.

1) Workspace Specification: Our goal is to find a representation of the workspace that is as simple as possible, yet still useful. Therefore, we decided on discretizing the otherwise continuous workspace. The simplest form is slicing the space in squares of equal size in the 2D case or cubes (voxels) in 3D, which is applied by the works mentioned in II-A. An alternative to this approach would be dynamically braking down occupied fields into smaller ones until a lower bound is reached (e.g. cell decomposition). This would result in a much more detailed map, but with the drawback of either

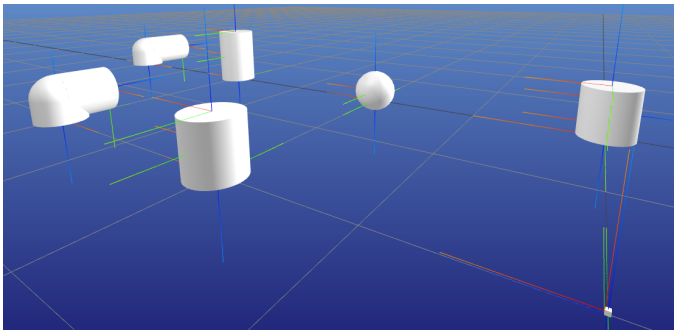


Fig. 1: Primitive modules in module database. From left to right: back row: l-link-30, i-link-30, end-effector, rotational joint; front row: l-link-20, i-link-20, base. Here the number denotes the length of the link, and the unit is millimetre (mm).

having to deal with a much larger matrix or having to use a different representation like a graph structure instead. For the sake of simplicity and under consideration of computational limitations, the first naive approach is still preferable for our purposes.

Besides a binary workspace that denotes reachability, we are also interested in the dexterity of each cell. As already mentioned in II-A, manipulability index (equation 1) of Yoshikawa is applied, which is available from the implementation of the *timor-python* toolbox by Külz et al. [4]. The manipulability index w is defined as follows:

$$w = \sqrt{det(JJ^T)} \tag{1}$$

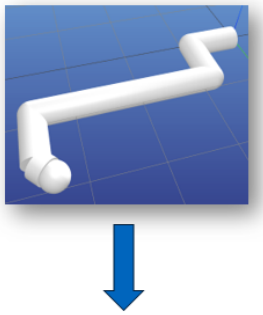
Since the implemented Jacobian J is of size 6xn for the 3D case, we need to adapt it to also work with 2D robots and workspaces. We therefore, only require the x, y and yaw (ϕ) derivatives in the Jacobaian J:

$$J^{'} = J[[x, y, \phi]] \tag{2}$$

$$w = \sqrt{det(J^{'}J^{'T})} \tag{3}$$

Still, errors arise for robots with less than 3 joints. Therefore, we decide on treating errors during the computation as edge cases and assign a value of -1. This allows us to keep those workspaces for the binary reachability case. The main advantages of using this index lies in its simplicity, but further research could explore alternative methods.

2) Robot Specification: Below, we explore how to build and represent modular reconfigurable robots (MRR) as input to our models. To simplify the task, we limit our MRRs to being sequential and loop-free. Furthermore, we only use robots capable of moving in the x-y-plane. Thereby, essentially dealing with 2D robots. We generate all of our robots randomly from a module database consisting of 7 modules (2 i-links, 2 l-links, 1 rotational joint, base, end-effector). They can be seen in Figure 1. Now that we defined the fundamental robot structures, we have to decide on how to represent them and feed them to the model. There are several methods of doing so. One of the most concise encodings is a binary encoding of



Binary Encoding:

Base	I20-Link	I30-Link	L20-Link	L30-Link	Joint	EEF
1	0	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	1	0	0
...						

Fig. 2: Exemplary binary encoding of a robot.

the used modules (see Figure 2). This approach omits explicit encoding of physical properties of the robot. A matrix is generated from the module sequence of the robot. With respect to the 7-module database, the binary encoding for any robot build from it consists of 7 columns with n rows, where n is equal to the number of modules of the robot. Since each robot starts with a base and ends with an end-effector, the encoding matrices of all robots have the same first and last row. Due to the implicit encoding of physical module properties, the network has to learn all parameters of each module just from the workspace data.

3) Workspace Generation: To ensure extensive training capabilities, we need to create a large dataset of a wide range of robots with different module configurations. Still, it is necessary to set some boundaries to ensure feasibility of the learning task. We choose to limit the robots to a maximum of 12 modules in sequence (at minimum 3). This equates to roughly 24 million different robots, with at least one joint choosing from our in section IV-A2 defined pool. In order to fit these robots in our discretized workspace, we define it to be from -3 to 3 in all dimensions. Furthermore, we limit the number of joints to 6.

The next question to answer is how much resolution the workspace should have. This trade-off between detail and computation time is especially significant for the 3D case, as the number of voxels increases cubically. As one can see in Figure 3 and 4 a resolution of 100 seems much smoother and better suited for our purposes. Additional increase in detail is not considered at this point since it would come at a significant cost of increased computation time.

With regard to 3D workspaces, the naive way of testing each cell for reachability by computing the inverse kinematics is not suitable for such high resolutions. The computation of a 10x10x10 workspace takes around 16s (Jacobian IK with maximum iterations set to 200, *timor-python* implementation).

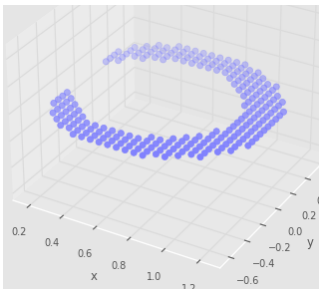


Fig. 3: Workspace with 80x80 resolution.

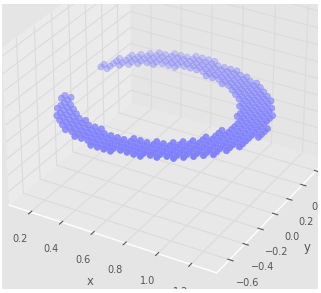


Fig. 4: Workspace with 100x100 resolution.

Further optimization (e.g. multiprocessing and iterative computation of inverse kinematics using the last configuration as a starting point) does not lead to significant enough improvements for it to be a viable option for a 3D workspace with a resolution of 100. Therefore, random sampling using forward kinematics with uniformly sampled random robot configurations is used. The idea is to generate randomly valid and non self-colliding configurations and compute for each of it the forward kinematics. We can then mark the corresponding cell in the workspace as reachable. For a 10x10x10 workspace we are able to generate this way 1000 samples in 0.1s or for a 101x101x101 workspace one million samples in less than 30s. This sampling method, while being less accurate with the workspace, is still more useful for our purpose of generating many workspaces in limited time.

Due to time constraints, we decide to only create a dataset for 2D robots. For the resolution, we choose 101x101 to easily center our robots in (0, 0). With the aforementioned sampling technique, we created a dataset of 1 million robots. Thereby sampling 20000 valid non self-colliding configurations (with a timeout at 300000 samples) for each robot. The generation of these workspaces takes roughly 12 hours, plus an additional 5 to 6 hours for robot generation itself. To save memory, we directly generate workspaces including the manipulability index, instead of a binary reachability indication. In the case of binary training, we treat any manipulability index unequal zero as a reachable cell. This is done during data loading while training.

We subdivide the large 1 million dataset into a 2k and a 100k dataset for the training. Thus, ensuring good coverage of different numbers of joints in the robots. The distributions of these sets can be seen in Tables I, II and III. We make the assumption that it is more difficult to find the workspace for robots with more joints, therefore tilting the distribution towards robots with higher joint counts. This assumption has to be kept in mind for the evaluation of our model (see Section V).

We choose to only tune hyperparameters on the 2k set to save on computation time. The 100k set is in turn used for training with the fixed hyperparameters of the 2k tuning phase. In the following chapter, we take a closer look at the model training.

TABLE I: Joint distribution over datasets. 1-DOF and 2-DOF samples are excluded for manipulability index prediction, since there are no valid values for those.

# Joints	1M	2k	100k (binary)	100k (regression)
1	35%	10%	10%	0%
2	35%	10%	10%	0%
3	20%	20%	20%	25%
4	8%	20%	27%	52%
5	2%	20%	20%	20%
6	0.3%	20%	3%	3%

TABLE II: Absolute distribution by number of modules

# Modules	1M
3	1
4	9
5	61
6	369
7	2101
8	11529
9	61476
10	217006
11	334168
12	373280

TABLE III: Absolute distribution by number of joints

# Joints	1M
1	352876
2	345144
3	201714
4	76889
5	20292
6	3085

(*LH*)

B. Learning Pipeline

After completing the data generation phase and on the basis of the chosen data input and output format, the architecture and module properties of the supervised learning pipeline can be defined in the next step. Figure 5 shows an overview of the proposed workflow, with the robot input illustrated on the left side and the resulting 2D workspace output visualized on the top right.

Based on the most influential components with regard to the performance of a supervised learning pipeline, the major design choices of the proposed implementation are two-fold: network architecture and supervision metrics for guided learning.

1) Network Architecture: As already established, in contrast to traditional robotic systems, the robot that is subject to our workspace identification task is of a modular nature. Ultimately, each of the considered robots consists of a set of sequentially assembled modules. Albeit representing the complicating factor for the underlying task, this intrinsic modular and sequential nature can be well exploited during the neural network design. Specifically, this interpretation of a robot input as a sequence of predefined modules essentially inspires the usage of a recursive type of network architecture.

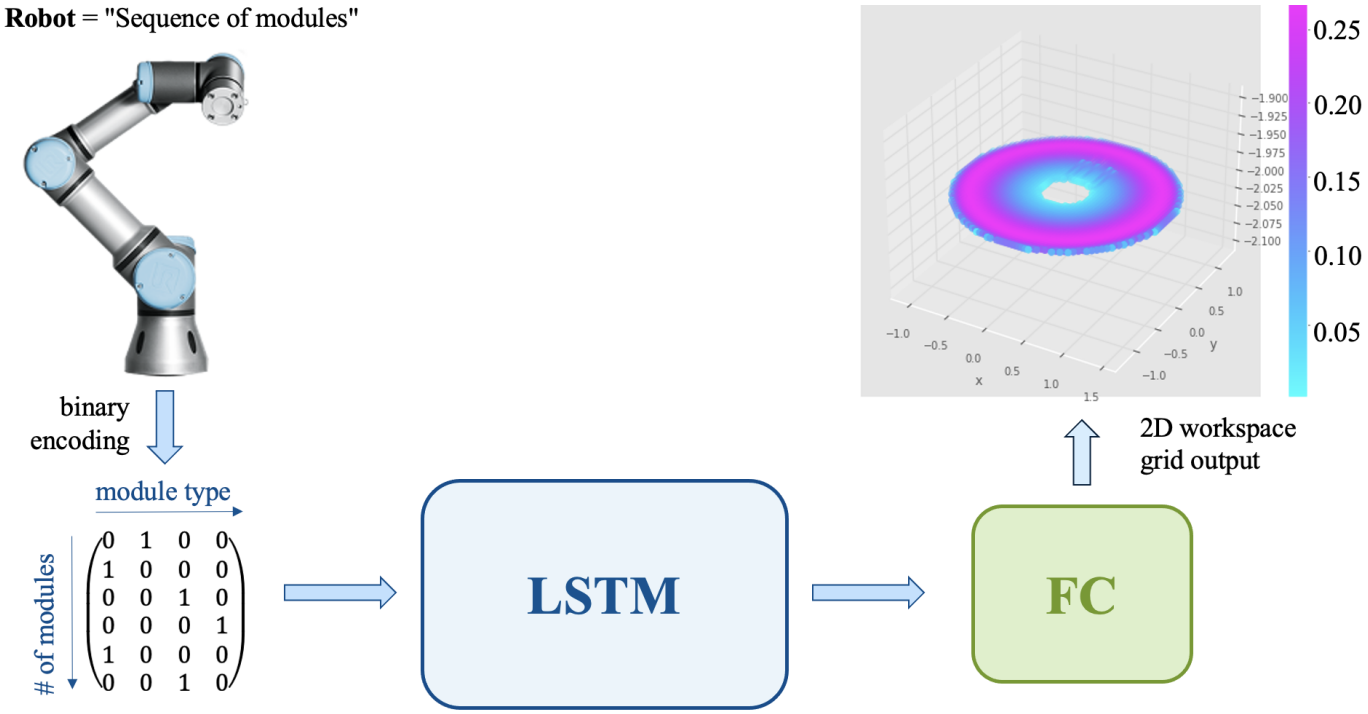


Fig. 5: Overview of the implemented supervised learning pipeline.

While there are several candidates for this sort of model, in the underlying solution a Long Short-term Memory (LSTM) [11] module is selected. This choice is mainly motivated in the fact that this network type comes with the promise of an increased stability in optimization behavior. In contrast, other recursive models like the classical Recurrent Neural Networks (RNN) or the more recently developed Transformers seem to be more difficult to train in certain cases. Reported reasons for this, among others, are problems with vanishing, exploding or unbalanced gradients [11], [15].

The selected LSTM can be seen in Figure 5 as being at the core of our proposed supervised learning method. It receives the binary encoding of a modular robot as an input, where each binary module vector is fed into one of the consecutive hidden states, thereby being able to handle varying-sized robot inputs. The resulting final hidden state as output of the LSTM module then needs to be further processed in order to be properly mapped to the same format as the desired 2D output workspace grid.

Several options are available for the design of this subsequent network module as well. For the problem at hand, a simple fully connected (FC) layer with a reshape operation is adopted (see Figure 5). Due to the resemblance of the 2D workspace output to an image format, deploying convolution operations was considered as well. During initial overfitting trials, however, this alternative was discarded due to the FC-layer performing better according to the expected behavior.

When it comes to the desired network behavior, an important part in supervised learning tasks is the definition of a suitable objective function providing targeted guidance for the model during its training. This aspect is covered in more detail in the following.

(*TB*)

2) Supervision Metrics: Regarding the optimization metrics, it is necessary to recall the distinction of the two problem cases considered in the context of the proposed method. For the binary reachability setting, on the one hand, the goal is to simply predict whether a certain position in the output workspace is reachable or not. This effectively renders it a binary classification problem. When it comes to the prediction of dexterity via the presented manipulability index, on the other hand, the task transforms into a type of regression, where the goal is to approximate a floating number at each grid position as best as possible.

According to this task distinction and the different optimization goals, two sets of supervision metrics need to be considered. Starting with the binary reachability, a multi-objective setting is deployed, where next to the loss function, two other metrics are optimized for. Following the idea of Porges et al. [2] to make use of confusion matrix values in order to evaluate prediction quality, we derive Positive Recall and Precision values as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

$$\text{Positive Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{5}$$

where TP, FP and FN indicate the number of True Positives, False Positives and False Negatives, respectively. Intuitively, the precision tracks the rate of actually reachable workspace

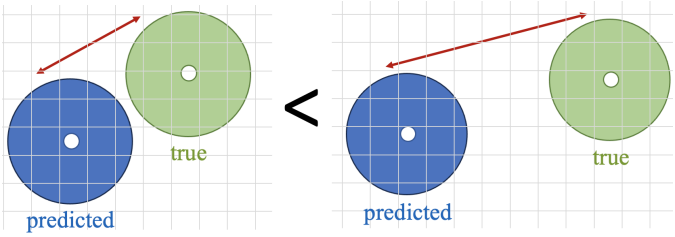


Fig. 6: Illustration of the intuition behind the inclusion of the Wasserstein set loss

positions among the ones that are predicted as such, while the positive recall measures how many of the reachable grid positions are truly detected. The goal during optimization is to maximize both values in order to ensure accurate workspace predictions. Note that the accuracy metric is deliberately excluded as a suitable performance indicator due to the fact that the output workspace grids are characterized by a strong class imbalance with mostly 1's being highly underrepresented, especially in the case of a 3D output workspace or for robots with less modules and joints. Therefore, trivially predicting only 0's would already imply large accuracy values.

Next to these binary classification measures, a training loss is minimized as:

$$\mathcal{L}_{Reachability} = (1-\gamma)\cdot\mathcal{L}_{EMD} + \gamma\cdot\mathcal{L}_{BCE} \tag{6}$$

where the total loss $\mathcal{L}_{total}$ is composed of a combination between the Binary Cross Entropy (BCE) loss typically employed in classification tasks and the Earth Mover Distance (EMD), which is a version of the Wasserstein loss commonly used for the comparison of two entire sets or distributions [16]. Here, γ is a scaling factor that is introduced as an additional hyperparameter during training.

The motivation behind the inclusion of a set loss is illustrated in Figure 6. While the BCE loss allows to encourage a high overlap between the predicted and true workspace, it can also happen that the prediction does not intersect with the ground truth at all. In this case, however, it is still desirable for the model to be able to assess the quality of the prediction. Considering the two cases from Figure 6, for instance, the output on the right should be assigned a higher loss than the left one due to an increased distance to the ground truth reachable workspace. This notion can be conveyed in the loss by including the EMD, which essentially measures how much energy is necessary to transform the blue set into the green one [16]. Therefore, by combining both the EMD and the BCE into the total reachability loss, the goal is to ensure both correct shape and spatial proximity between predicted and true workspace.

Optimal configurations for hyperparameters are obtained by using the Optuna library [17]. Since γ renders the loss value variable, the combined loss is not used as the main objective for optimization. Rather, the aforementioned positive recall and precision are considered as the two primary objectives. Note that Optuna returns sets of which the objectives are on the pareto front [18] as the best trials during multi-objective optimization. In this case a trial might have either higher precision or higher positive recall. We argue that precision is of more importance than positive recall within the best trials since it is reasonable that the network can not detect the singular points. However, if the model predicts a larger workspace that has 100% coverage of the true one as indicated by a maximum positive recall value of 1, the result can be misleading in the sense that the capabilities of the robot under examination would be represented as overestimated. Certain task that the manipulator is not able to perform could be wrongfully included in the workspace map, which should ultimately be avoided.

TABLE IV: Hyperparameters for both two tasks

	Binary Reachability	Manipulability
Layer Number	1	1
Hidden Size	900	300
Learning Rate	$4.014030472773061\times 10^{-3}$	$7.023522566033369\times 10^{-3}$
Batch Size	32	16
Dropout Rate	0.5	0.5
Epochs	50	40
Loss Scaling Factor γ	0.9	-

Finally, concerning the optimization task of learning to predict manipulability values, the regression loss is chosen as follows:

$$\mathcal{L}_{Manipulability} = \mathcal{L}_{MSE} \tag{7}$$

where subject to minimization is the Mean Squared Error (MSE) between the predicted and ground truth manipulability values. In this case, the MSE loss is utilized as the only criterion for the regression task. The hyperparameters for the final training and testing for both cases are listed in Table IV. The result values as well as the final visual workspace outputs for both the considered optimization tasks are shown and evaluated in the upcoming section.

(*TB, XC*)

V. Evaluation

After learning pipeline is established, one model is fitted by the samples labeled with binary values, and the other is fitted by those labeled with manipulability index. In this section, the results of prediction of binary reachability are illustrated and discussed in the first subsection, and the second subsection is regarding to the prediction of manipulability.

(*XC, TB*)

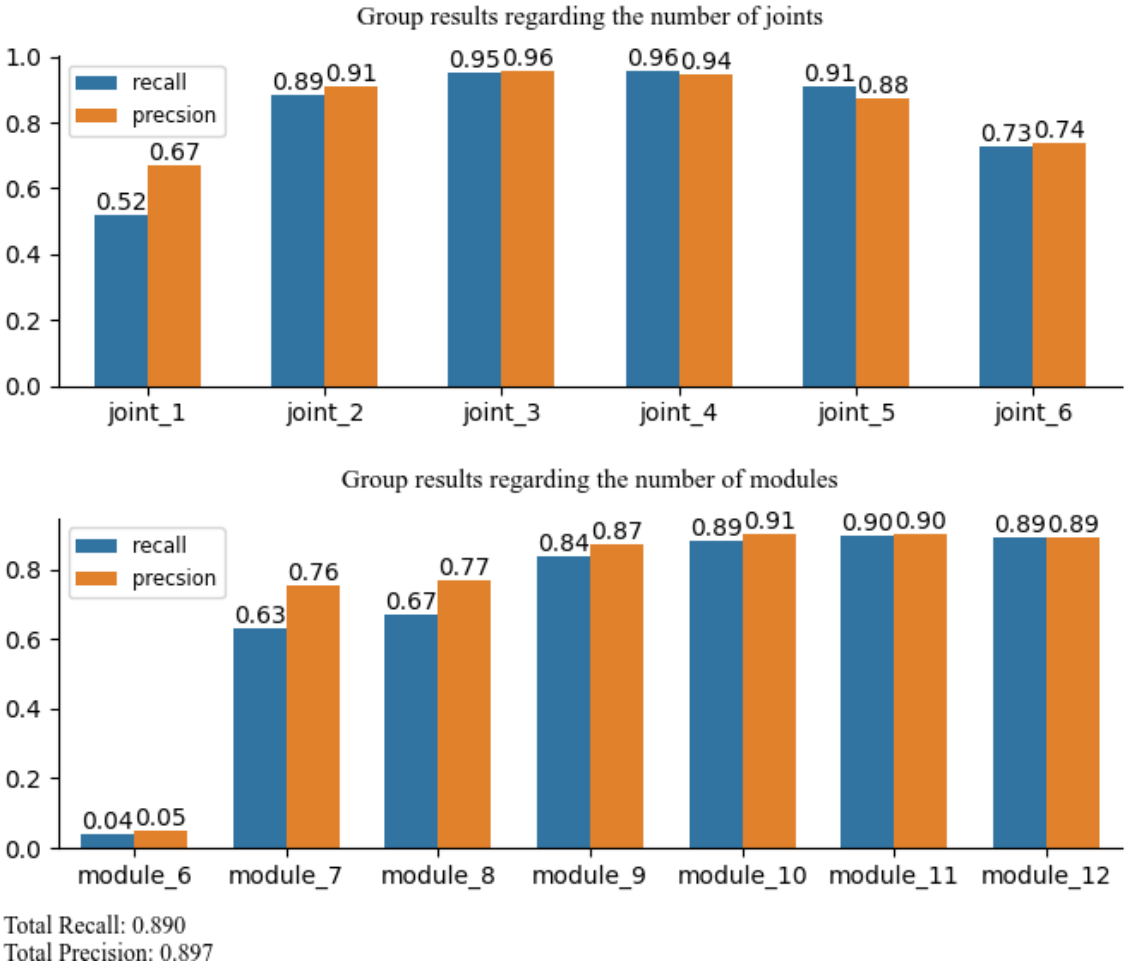


Fig. 7: Group results of predicting binary reachability regarding number of joints and modules

A. *Results of Predicting Binary Reachability*

After training, the model reaches 89.0% for positive recall and 89.7% for precision on the test dataset containing 10,000 samples[1], and the inference time is 9.421 seconds on a Graphics Processing Unit (GPU) in total. Both precision and recall achieve quite high values, and the precision, as desired, slightly higher than the recall. The performance in terms of efficiency of inference is outstanding compared with the conventional methods mentioned in II-A.

To obtain better insights into the results, we divided samples according to their own number of joints and modules, respectively, and computed positive recall and precision for each group. The results are illustrated in Figure 7. Furthermore, the visualization of the predicted reachable workspace are demonstrated joint-wisely in the third row of Figure 8. For robots with different numbers of joints, those with 3 and 4 DOF result in excellent metrics, while the performance is downgraded as the number of joints increases. However, the model performs considerably worse than expected on robots with a single joint. For robots with varying modules[2], a number of modules higher than 10 turns out to be beneficial for the predictive performance.

Since each robot is randomly assembled without any preference on modules, a robot with less modules might have less joints and vice versa. To investigate whether both number of joints and modules have impact on the performance or not, we further analyzed the results of two groups of robots by controlling variables, which are demonstrated in Table V. As a result, the value distribution of metrics with respect to the number of joints and modules is quite similar to the overall results in Figure 7. Therefore, we conclude that the performance of the model is distinguished by number of joints and number of modules as well.

In terms of reasons for unsatisfactory performance of certain

[1]The batch size of the test dataloader is set to 1. The values of both the precision and recall metrics are improved with an increased batch size since the labels of robots with higher reachability have more TPs, which induces a bias in the results. Nonetheless the loss remains the same.

[2]Note: the test dataset does not contain robots with 3, 4, or 5 modules

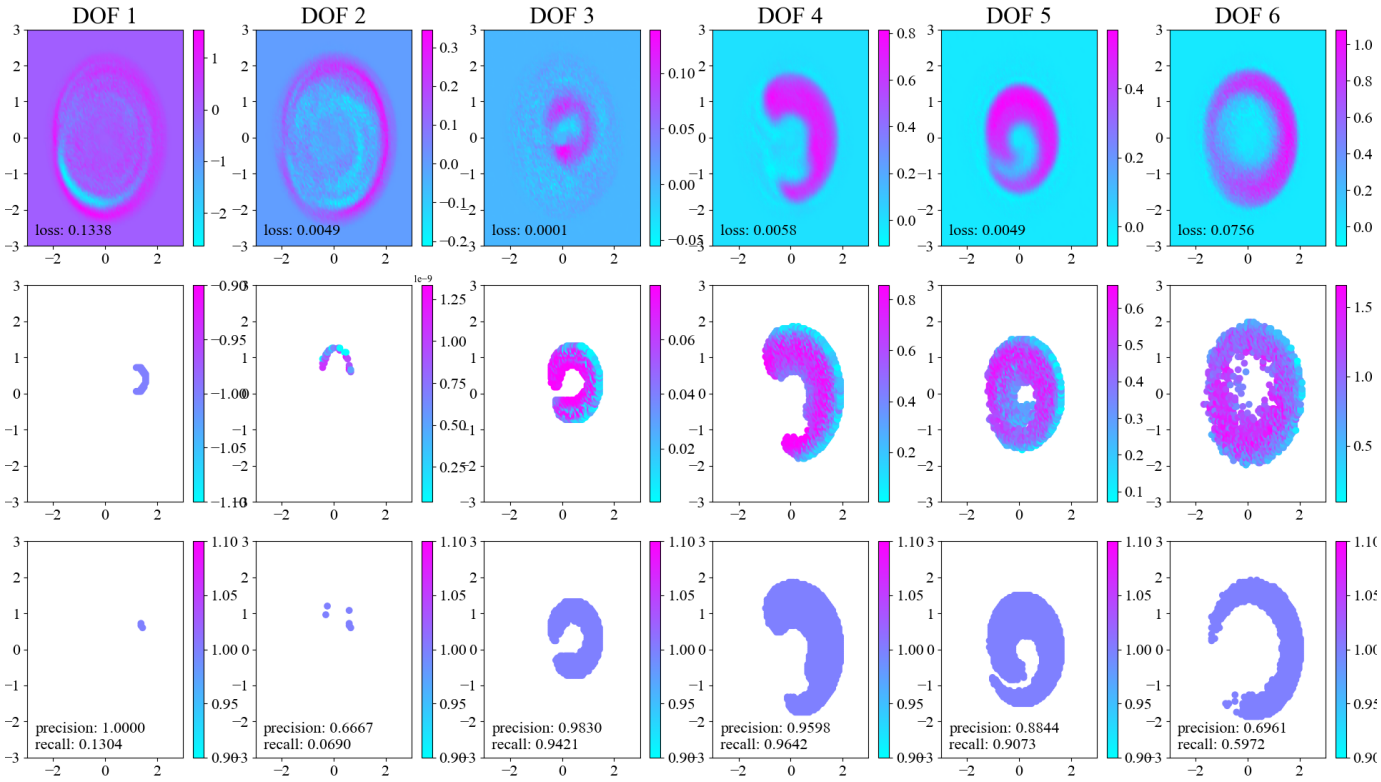


Fig. 8: Visualization of predicted and true workspace map. First row: predicted manipulability map; Second row: ground truth; Third row: predicted binary reachability map. On the bottom left of each sub-figure are the metrics of the corresponding samples.

groups, there are two main aspects. On the one hand, the data volume of robots with 6 joints, which have higher dexterity and can yield more complex workspace shapes in comparison to manipulators of smaller DOF, is insufficient since the 6-DOF samples only take up the least portion (3%) of the 100K dataset (see Table I). By comparison, the number of considered assemblies are increased on purpose along with increase of the number of modules (See Table II). As a result, the robots with more modules, of which the shapes of workspace are more diverse, achieve better results. On the other hand, the robots with a single joint or with less than 8 modules suffer severe class imbalance in the labels (mostly less than 100 positive 1's within 10201 grids), which makes it more difficult for the model to detect the positive cells.

(*XC, TB*)

B. Results of Predicting Manipulability Index

Based on a similar training procedure as for the binary reachability prediction task, in the case of the manipulability index regression task, the model yields 0.0076 as loss value in 9.593 seconds on GPU. The test samples are grouped in the same manner as in the previous case for the computation of the MSE loss, and the results are shown in Table VI & VII. However, according to the visualization in Figure 8, we realized that the MSE loss is not a proper metric for evaluation on this task. As illustrated, the samples with 4 and 5 DOF

TABLE V: Results of robots with either the same number of joints or the same number of modules. The darker is the color, the higher is the value.

# Joints	# Modules	**Precision**	**Recall**
1	12	0.62	0.49
2		0.90	0.89
3		0.95	0.95
4		0.94	0.95
5		0.86	0.90
6		0.71	0.68
2	7	0.78	0.65
	8	0.83	0.83
	9	0.88	0.86
	10	0.92	0.90
	11	0.92	0.89
	12	0.90	0.89

should be assigned with better results since the predicted region with high manipulability index is highly overlapped with the ground truth. However, the 3-DOF sample receives the best result, in which the boundary of the region with high manipulability index is blurry. Even the prediction of the sample with 2 DOF, which does not seem reasonable, achieves the same result as the one with 5 DOF. Note that the model is not trained with the samples with 1 or 2 DOF (see IV-A), and the maps here are merely trials for inference.

TABLE VI: MSE loss regarding groups of joints when predicting Manipuability index

# Joints	Loss
3	0.0012
4	0.0049
5	0.017
6	0.049

TABLE VII: MSE loss regarding groups of modules when predicting Manipuability index

# Modules	Loss
7	0.00055
8	0.00044
9	0.00081
10	0.0021
11	0.0055
12	0.011

The reason for the impropriety of MSE loss could be that it measures the square of absolute error to the ground truth and does not consider the relative error. For instance, the highest predicted manipulability values are 0.125 and 1.0 in the samples with 3 DOFs and 6 DOFs, respectively, while the truths are 0.07 and 1.6, so the errors are 0.003 and 0.36 here. However, the error rate of the prediction at this point for the 3-DOF sample is close to 100%, while the error rate for 6-DOF sample is 60%. To tackle this problem, Mean Absolute Percentage Error (MAPE) [19] can be an alternative, which calculates the average percentage difference between the predicted and true values and is useful for measuring the relative error. Nonetheless, the current model can still deliver meaningful results by using the MSE as objective for learning since a perfect model does have 0 MSE-loss, and the overall loss, 0.0076, is already in a very low level. Furthermore, from the visualization in Figure 8, apparent correlation between the prediction and the ground truth of 3-DOF robots can be observed, and the regions with locally high manipuability indices can be well distinguished for the robots with more than 4 DOFs.

(*XC, TB*)

VI. Conclusion and Future Work

Overall, the elaborations at hand demonstrate that it is indeed possible to capture the non-linear relationship between the characteristics of a modular robot assembly and its reachable workspace using a neural network. On a first 2D setting of this workspace identification task, the proposed implementation based on a supervised learning scheme delivers a final model that exhibits highly precise results both visually and quantitatively, especially on MRRs with a higher number of DOF.

The in-depth analysis of our final results also uncover potential improvements regarding simpler robots with only one or two joints. One possible solution is applying rate constraints on each group to enforce the precision and positive recall reach a certain threshold [20] in the optimization pipeline. Additionally, while the manipulability output of our network does deliver suitable and intuitive results, it still lacks and interpretable quantitative value. Some extensions of the current network implementation are needed in order to enforce it to deliver a useful manipulability index that is restricted to positive values only. Furthermore, alternative methods of quantifying dexterity in the workspace can be explored in the future.

With respect to the training data, future research should ensure a more equal distribution of samples, especially regarding the 6-joint case. Apart from that, there are several extensions to the current solution that could be considered in the future. A first step would be to remove the restriction of robot movements to only a 2D plane and to tackle the problem of workspace identification for the 3D space in its entirety. This elevates the complexity of the task considerably, potentially implying a necessary reevaluation of certain aspects of the proposed method, such as the network architecture. As briefly discussed in the respective subsection IV-B, other options exist for the choice of model backbone. Transformers are a currently popular choice in problems of sequential nature but require more exploration with regards to data requirements and fine-tuning. Besides that, different structures like a Graph Neural Network (GNN) could be considered as a direction for future research as well. Furthermore, improvements to the storage manner and representation of the workspace could be necessary for 3D, since comparably even more cells are labeled as 0.

On top of that, the selected robot representation can be subject to advancement as well. Especially, the binary encoding of the module sequence is initially chosen mainly for simplicity reasons. However, other alternatives that do not rely on a predefined set of robot modules but instead also consider physical properties in an intrinsic manner promise to elevate the quality and capability of the neural network model even further.

(*TB, LH, XC*)

REFERENCES

[1] Y. Guan and K. Yokoi, "Reachable space generation of a humanoid robot using the monte carlo method," 10 2006, pp. 1984–1989.

[2] O. Porges, R. Lampariello, J. Artigas, A. Wedler, C. Borst, and M. A. Roa, "Reachability and dexterity: Analysis and applications for space robotics." German Aerospace Center (DLR), 2015.

[3] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: Representing robot capabilities," 12 2007, pp. 3229 – 3236.

[4] M. A. Jonathan Külz, Matthias Mayer, "Timor python: A toolbox for industrial modular robotics," in *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2023, to appear.

[5] D. Kee and W. Karwowski, "Analytically derived three-dimensional reach volumes based on multijoint movements," *Human factors*, vol. 44, pp. 530–544, 2002.

[6] O. Porges, T. Stouraitis, C. Borst, and M. A. Roa, "Reachability and Capability Analysis for Manipulation Tasks," in *ROBOT2013: First Iberian Robotics Conference*, ser. Advances in Intelligent Systems and Computing, M. A. Armada, A. Sanfeliu, and M. Ferre, Eds. Cham: Springer International Publishing, 2014, pp. 703–718.

[7] T. Yoshikawa, "Manipulability and redundancy control of robotic mechanisms," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 1004–1009.

[8] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann, "Manipulability analysis," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 2012, pp. 568–573.

[9] J.-O. Kim and K. Khosla, "Dexterity measures for design and control of manipulators," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, 1991, pp. 758–763 vol.2.

[10] C. Benka, C. Gross, R. Gupta, and H. Lipson, "Direct robot configuration space construction using convolutional encoder-decoders," 2023.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] Y. Zhu, H. Zhang, G. Zhou, Z. Liang, R. Lyu, and Y. Liu, "A dynamics modeling method of manipulator based on long short term memory neural network," in *2021 4th International Conference on Mechatronics, Robotics and Automation (ICMRA)*, 2021, pp. 102–107.

[13] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, p. 43–52, 2007.

[14] J. Liu, X. Zhang, and G. Hao, "Survey on research and development of reconfigurable modular robots," 2016. [Online]. Available: https://journals.sagepub.com/doi/epub/10.1177/1687814016659597

[15] L. Liu, X. Liu, J. Gao, W. Chen, and J. Han, "Understanding the difficulty of training transformers," 2020.

[16] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[17] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[18] J. Legriel, C. Le Guernic, S. Cotton, and O. Maler, "Approximating the pareto front of multi-criteria optimization problems," in *Tools and Algorithms for the Construction and Analysis of Systems*, J. Esparza and R. Majumdar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 69–83.

[19] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, 2016, advances in artificial neural networks, machine learning and computational intelligence. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216003325

[20] A. Cotter, H. Jiang, S. Wang, T. Narayan, S. You, K. Sridharan, and M. R. Gupta, "Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals," *Journal of Machine Learning Research*, 2019.