2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Inverse Kinematics of Robotic Manipulators Using a New Learning-by-Example Method

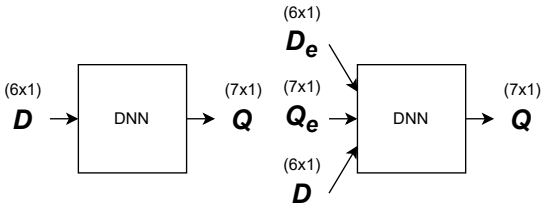
Jacket Demby's, *Student Member, IEEE,* Ramy Farag, *Student Member, IEEE,*
and Guilherme N. DeSouza, *Member, IEEE*

***Abstract*— Inverse Kinematics (IK) is one of the most fundamental challenges in robotics. It refers to the process of determining the joint configurations required to achieve the desired position and orientation (pose) of a robot end-effector. Although numerous Data-Driven (DD) IK solvers have demonstrated encouraging results, they have not achieved the same accuracy when compared to other IK methods for complex robot configurations (e.g., numerical methods for higher Degrees of Freedom (DoF)). In this work, we propose a new Learning-by-Example method, and show that such a scheme considerably improves the IK learning results when compared to other DD learners. In our approach, the network input incorporates an example of joint-pose pair along with the query pose to predict the desired robot joint configuration. We show that the example joint-pose pair does not need to be too close to the query – i.e. example and query can be as far as 20 degrees apart in the joint configuration space. Furthermore, we investigate the utilization of residual and dense skip connections in Multilayer Perceptron for DDIK solvers and employ the resulting networks for two redundant robotic manipulators: a 7-DoF-7R commensurate robot and a 7-DoF-2RP4R incommensurate robot. Our experimental results show that the resulting DDIK solver can reliably predict IK solutions with accuracy better than $1mm$ in position and $1deg$ in orientation.**

I. Introduction

Programming serial robotic manipulators involves knowing two main functions, namely: the Forward (FK) and Inverse Kinematics (IK). The FK function (denoted as $f_{FK}(Q)$) defines a mapping $f_{FK} : \mathbb{R}^n \rightarrow SE(3)$ from the joint configurations $Q \in \mathbb{R}^n$ to the robot end-effector poses $D \in SE(3)$. Computing $f_{FK}(.)$ is relatively simple using the Denhavit-Hartenberg (D-H) [1] or the Elementary Transform Sequence (ETS) methodologies [2], [3]. As the name implies, the IK function (denoted as $f_{IK}(D)$) defines the inverse mapping to the FK. Also, while an $f_{FK}(.)$ solution is uniquely defined, an IK solution may have an infinite number of solutions, while it can also be difficult or even impossible to derive in a closed-form manner (e.g. for complex robot configurations with high Degrees of Freedom (DoF)) [4]. Due to the aforementioned reasons, the IK problem has been the subject of much research. Several IK methods have been proposed over the years and can be divided into the following categories: Analytical, Geometrical, Numerical, Optimization-Driven (e.g. soft or evolutionary computing), Data-Driven, and Hybrid [5], [6].

All authors are with the Vision-Guided and Intelligent Robotics (ViGIR) Laboratory in the Department of Electrical Engineering and Computer Science (EECS), University of Missouri-Columbia, Columbia, Missouri, 65201. (emails: udembys@mail.missouri.edu; rmf3mc@missouri.edu; desouzag@missouri.edu)



(a) Learning without Example (b) Learning-by-Example

Fig. 1: Illustration of the input-output mapping schemes investigated in this paper for learning IK problems using Deep Neural Networks (DNN). In: **(a)** A traditional query pose D to joint Q mapping, versus **(b)** The proposed Learning-by-Example, where the example pair (D_e, Q_e) is provided to guide the learning of query pose D to joint Q mapping. In this work, $D \in \mathbb{R}^6$ and $Q \in \mathbb{R}^{nDoF=7}$.

Analytical and Geometrical IK methods provide a globally optimal solution, and in most cases they can efficiently and reliably anticipate situations with multiple solutions. However, these methods are only applicable to simpler robot models with few DoFs [7], [8], as it is well known that closed-form solutions become more complex and difficult to derive for non-trivial robotic designs with higher number of DoFs. Furthermore, when one exists, analytical expressions must be uniquely derived for each robotic design.

On the other hand, IK methods based on numerical (e.g., pseudoinverse-based [9], selectively damped least-squares [10], etc.) and Soft-computing (e.g., genetic algorithm [11], particle swarm optimization [12], etc.) approaches iteratively update the vector of joint configurations until convergence using nonlinear search techniques. Unfortunately, these methods are sensitive to various user-defined design parameters; they are likely to get locked in local minima; and they may be subject to high computational costs and long convergence times [7], [8]. For example, the performance of numerical IK methods may be affected by initial joint configurations, chosen gains, and predefined pose tolerances as stopping criteria. Furthermore, the outcomes from Numerical IK approaches are highly affected by null space reference postures in the case of redundant manipulators [13], [14]. In the same sense, the performance of Soft-Computing methods is significantly influenced by the design of the fitness function, while Data-Driven IK (DDIK) methods, which employ machine learning and/or deep learning techniques, depend on the quality of previously generated datasets to learn a *one-to-one* or *one-to-many* mapping between the desired/queried end-effector poses and corresponding joint configurations – i.e. one pose to one joint configuration [4], [15] or one pose to multiple joint configurations [16], [17]. Finally, any of

these IK methods can be combined to create a Hybrid IK method [18], [19], however, they have been shown to carry over some of the same flaws of their parents' methods.

In recent years, DDIK solvers have gained increasing interest from the robotics community because once the training has been accomplished to a significant level of accuracy, DDIK solvers can more quickly and efficiently compute IK solutions when compared to other approximate methods – e.g. without relying on iterative processes. However, they still fall short to produce accurate IK predictions in real-world IK applications that require sub-millimeter position and sub-degree orientation accuracy [4], [20].

In this work, we focus on learning *one-to-one* IK mappings with a new Learning-by-Example scheme that incorporates joint-pose examples into the network training (Figure 1). Moreover, motivated by the success of ideas such as residual and dense skip connections in other domains [21]–[23], we integrate the aforementioned types of connections into the MLP architecture, as depicted in Figure 2 to further improve the learning of IK solvers. Our findings reveal that the resulting networks trained with the proposed Learning-by-Example model well outperformed their counterparts trained without the example pairs, and that the MLP architectures with residual and dense skip connections are even more capable of learning-by-example. So, in that sense, the main contributions of this research are as follows:

- we introduce a Learning-by-Example strategy, that uses an example pair with pose and joint configurations alongside with the query pose to predict the associated joint configuration. Different from the work by Xing et al. in [24], which utilizes sequences of previous and current poses and previous joints to predict current joint in the same sequence (path), we demonstrate that the sole reliance on example joint-pose pairs in the robot workspace yields high(er) accuracy in the investigated DDIK solvers.
- we incorporate residual and dense connections within a baseline MLP architecture – with and without Learning-by-Example – to approximate the Inverse Kinematics (IK) of both commensurate and incommensurate redundant robotic manipulators [25], [26].
- we demonstrate that sample joint-pose pairs do not need to be at close proximity to the query joint-pose pair; in fact, these can be as far as 20 degrees in the joint configuration space to still yield sub-millimeter accuracy in pose.
- we conduct a comprehensive performance comparison between three models: a Plain MLP, a Resnet-like MLP (RMLP), and a Densenet-like MLP (DMLP), evaluating their efficacy in predicting an IK solution when trained with and without an example joint-pose pair. The evaluation primarily focuses on the accuracy of the reconstructed end-effector Cartesian poses (positions and orientations) from the predicted joints.
- we compare the investigated DDIK solvers to three numerical solvers that differ based on their inverse Jacobian methods: Selectively Damped Least Squares (SD) [10], Singular Value Filtering (SVF) [9], and Mixed Inverse (MX) [27].

The remaining of this paper is organized as follows: Section II provides a quick survey on the recent milestones of DDIK solvers for robotic manipulators. Section III details the main network architectures, the generated datasets, and the network input schemes employed in this work. In Section IV, we present and analyze the results of all our experiments. And, in Section V, we conclude this paper with insights on current limitations and future work directions.

II. Related Work

As previously noted, DDIK solvers, also known as Learning-based IK methods; applied to serial robotic manipulators, have gained lots of interest from the robotics community in recent years. Since the seminal work of Guez and Ziauddin [28], who employed a Multilayer Perceptron (MLP) to solve the IK of two commensurate manipulators having 2- and 3-DoF; various DDIK solvers have been introduced ranging from traditional Machine Learning (ML) techniques (e.g., Support Vector Machines [29]) to modern deep learning approaches (e.g., Transformers [24]). About these recent developments, Dalmedico et al. [15] approached the IK learning problem of a 4-DoF commensurate manipulator with a shallow MLP. By constraining the workspace and employing only the position as input to the network, they were able to predict IK solutions with less than $1cm$ average reconstructed position errors. Volinski et al. [30] employed various Artificial and Spiking Neural Networks for the IK learning problem of a 6-DoF commensurate manipulator. In their work, they constrained the robot's workspace during the dataset generation and exclusively trained the networks using a pair of end-effector positions apart from each other within $3cm$, while not taking into account the orientations. Dembys et al. [4], [6] conducted a comparative IK study based on four commensurate and incommensurate robots (4- to 7-DoF), three Neural Networks (NN) architectures (all-joints-MLP, individual-joints-MLP, and individual-joints-ANFIS) and two dataset types (constrained and unconstrained in the robot workspace). They employed both position and orientation as inputs when training the networks. Their conclusion highlighted that the examined NN architectures displayed significant errors compared to numerical methods, were not a fruitful approach as DDIK solvers, and needed more investigations. Ames et al. [17] proposed a DDIK solver that leverages the recent advances from the Normalizing Flows domain, namely: IKFlow. They employed a generative Conditional Invertible Network (cINN) in an attempt to learn a $one-to-many$ mapping to accommodate multiple IK solutions. They employed ten robots with various complexities (between 4- and 10-DoF), very deep cINN architectures (between 108 and 240 layers by considering the coupling layer width and the number of coupling layers), and were able to reach around $3mm$ average pose and $1deg$ average orientation accuracies of all the investigated robots. More recently, Kim and Perez [19] employed a hybrid IK solver combining an autoencoder architecture and a Numerical IK method for a

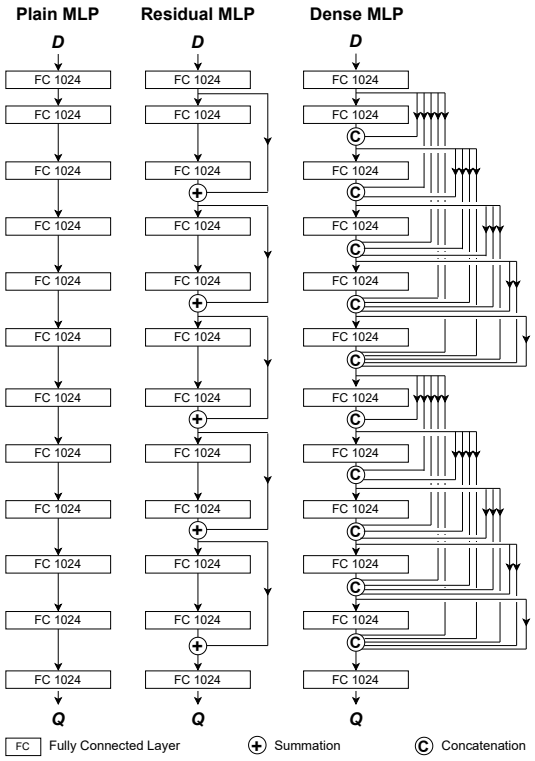


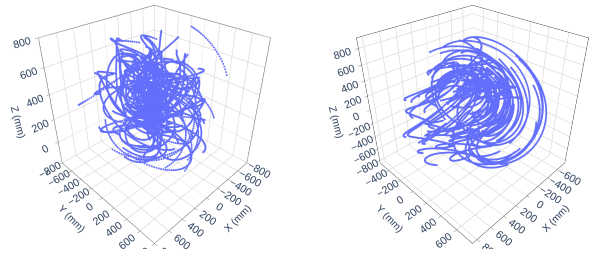
Fig. 2: Example Deep Neural Networks investigated as DDIK solvers; **left:** Plain Multilayer Perceptron (MLP) as a baseline, **middle:** Residual-like MLP (RMLP), and **right:** Dense-like MLP (DMLP). *FC 1024* refers to a Fully Connected layer having 1024 neurons.

7-DoF robotic arm. The Numerical IK solver was used to refine the network predictions by employing them as initial joint configurations. Xing et al. [24] attempted to learn the IK of multiple robots with a Transformer architecture. The network input was made of a sequence of previous poses, previous joints, and current poses to predict a sequence of current joints. However, their dataset was made of only robot structures with at most 6-DoF ($\leq$ 6-DoF) and they reported results for only 3-DoF robots. Drawing inspiration from the success of skip connections-based NN architectures, particularly ResNet [21] and DenseNet [22] in domains such as Image Classification and Image Recognition, we investigate two MLP-based architectures. The first is constructed with residual skip connections, denoted RMLP, and the second utilizes dense skip connections, referred to as DMLP. These architectures are then employed to approximate the *one-to-one* IK of robotic manipulators.

III. Proposed Methodology

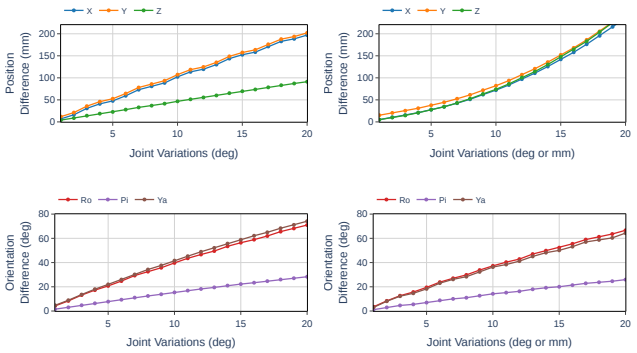
A. Dataset Generation

To the best of the authors' knowledge, there are no well-accepted and publicly accessible benchmark datasets to train DDIK solvers for robotic arms, in contrast to various other Deep Learning-related challenges (e.g., Image Classification [31], Image Segmentation [32], Lidar Odometry [33], etc.). Most of the currently available DDIK Solvers are trained based on different dataset generation schemes that rely on the D-H methodology and the f_{FK} to create a mapping



(a) 7-DoF-7R robot (b) 7-DoF-2RP4R robot

Fig. 3: Illustrations of the generated trajectory-based datasets for the investigated robots with a joint variation $v = 1^o$, visualized here for 100 trajectories of 100 samples each. The number of samples has been reduced in the plots for visualization purposes (better viewed when zoomed in), however all the generated datasets had 1.000.000 samples.



(a) 7-DoF-7R robot (b) 7-DoF-2RP4R robot

Fig. 4: Average difference in X, Y, Z, Ro, Pi, and Ya between the example pose (in the joint-pose pair, or D_e and Q_e) and the query pose D versus joint variations v. This average was collected for each robots investigated and applied to during the generation of the training/testing datasets.

between the input poses D and output joint configurations Q. In this work, we generated trajectory-based datasets as depicted in Figure 3. For each investigated robot, the corresponding dataset was made of 10.000 trajectories of 100 steps each. Similar to [24], we randomly sample an initial joint configuration for each trajectory from the full unconstrained robot workspace and acquire the next joint configuration in the trajectory by deviating the joint state within a revolute joint variation $\pm v \in [1deg, 20deg]$ or a prismatic joint variation $\pm v \in [1mm, 20mm]$ of the current joint values. In total, each of the generated datasets had 1.000.000 samples. We decided to set 80% for training, 10% for validation, and 10% for testing. Figure 4 shows the resulting average distances between the example pose D_e and the pose of interest D based on the joint variations chosen when generating the dataset. The goal behind varying the joint variation v was to show how far the example joint-pose pair needs to be from the query for the investigated network to provide reliable IK solutions.

B. Network Architectures

We investigated different types of MLP-based NN architectures — Plain MLP (Figure 2, left), Residual MLP (Figure 2, middle) denoted RMLP, and Densely Connected MLP (Figure 2, right) denoted DMLP — for the IK learning problem. The primary distinctions between RMLP and DMLP stem from the specific skip-connections utilized and

TABLE I: D-H Parameters of the commensurate and incommensurate serial robots used for the experiments. The angles θ and α are expressed in *deg*, while the variables d and a are expressed in *mm*.

i	θ	d	a	α
1	θ_1	333	0	0
2	θ_2	0	0	-90
3	θ_3	316	0	90
4	θ_4	0	82.5	90
5	θ_5	384	-82.5	-90
6	θ_6	0	0	90
7	θ_7	107	88	90

(a) 7-DoF Panda arm

i	θ	d	a	α
1	θ_1	0	0	90
2	θ_2	0	250	90
3	0	d_3	0	0
4	θ_4	0	0	90
5	θ_5	140	0	90
6	θ_6	0	0	90
7	θ_7	0	0	0

(b) 7-DoF GP66+1 arm

the arrangement of these connections within the network architecture. In RMLP, summation-based skip connections are employed and the network is made of stacked residual blocks. Each residual block is chosen to have two fully connected (FC) layers, where the first FC layer is followed by a ReLU activation and another FC layer; the output this second FC layer is summed with the output from the previous residual block before being passed to another ReLU activation. In DMLP, skip connections based on concatenation are utilized, and a transition layer is employed to decrease the dimensionality following the concatenation. Each dense block is chosen to have five FC layers where the input to a FC is made of a concatenation of the outputs of all the previous layers. By choosing these two skip connection-based networks, the goal was to investigate if adding shortcut connections to a baseline MLP would improve its performance. That is, after choosing the baseline MLP model, RMLP and DMLP models were designed to have the same number of layers for fair comparison as depicted in Figure 2 and reported in Section IV.

C. Loss Function

All the investigated networks were trained by minimizing the Mean Squared Error (MSE) loss denoted here by L_Q as shown in Equation 1:

$$L_Q = \frac{1}{N}\sum_{i=1}^{N} \left\| Q_i - \hat{Q}_i \right\|_2^2 \tag{1}$$

where $\hat{Q}$ and Q are respectively the predicted and ground truth joint configurations, and N is the total number of samples in the training set.

D. Evaluation Metrics

The network performances were evaluated by finding the reconstruction pose errors defined as the difference between the homogeneous transformation $\hat{T} = f_{FK}(\hat{Q})$ and $T = f_{FK}(Q)$ as described bellow:

$$\begin{aligned} T_{error} &= T * \hat{T}^{-1} \\ T_{error} &= \begin{bmatrix} R_{error} & t_{error} \\ \mathbf{0}_{3\text{x}1} & 1 \end{bmatrix} \end{aligned} \tag{2}$$

where t_{error} is the position error and R_{error} corresponds to the rotation error that has been converted to its corresponding Roll-Pitch-Yaw (RPY) orientation representation for all the results we reported in Section IV.

IV. Experimental Results and Discussion

In this section, we present a series of experiments aimed at showcasing the benefits of Learning-by-Example and the integration of residual and dense connections in an MLP-based DDIK solver. Specifically, our proposed architectures were applied to solve the IK learning problem of two 7-DoF redundant manipulators: a 7-DoF-7R commensurate Panda robot arm and a 7-DoF-2RP4R incommensurate GP66+1 robot arm. Table I provides their respective D-H parameters. We implemented all the investigated models using PyTorch and have made the code publicly accessible on the GitHub repository associated with the paper[1] to allow reproducible research. All the reported results in this section are an average of 10 different runs of the investigated networks with different random seed values. We also evaluated the performance of the investigated DDIK solvers against three established numerical methods, each employing a different approach for the inverse Jacobian calculation: Selectively Damped Least Squares (SD) [10], Singular Value Filtering (SVF) [9], and the Mixed Inverse (MX) [27]. We refer the readers to [9], [10], [27] for more details on these methods.

A. Architectures and Parameters

As mentioned before, we chose the Plain MLP performing a *one-to-one* mapping between the pose and joint configuration spaces (Figure 1a) as the baseline model. The MLP was chosen to have 12 hidden layers with 1024 hidden neurons, the optimizer type was Adam, the learning rate was set to 0.001, and the activation function at all layers was Rectified Linear Unit (ReLU) — except for the output layer which did not have an activation function. The input poses were normalized between -1 and 1 while the output joint values were kept to their generated values in radians. The baseline MLP model was trained for 1000 epochs with the L_Q loss (as shown in Equation (1)) and a batch size of 128 while utilizing an early stopping strategy with a patience tolerance of 100 epochs. For a fair comparison, we maintained the same training strategy described above for the training of all the investigated models with both input schemes depicted in Figure 1.

For the numerical solvers, we set the number of iterations to 500, the attenuation factor to $\alpha = 1$ and the target pose error to $1mm$ in position and $1deg$ in orientation. We also used the joint values from the example joint-pose pair to be the initial joint configuration when solving for the query pair.

B. Influence of the input schemes

We propose to employ a Learning-by-Example strategy to train the investigated networks with an example pair of joints Q_e and pose D_e in addition to the pose D of interest to predict the corresponding joints Q. We argue that the provided example pair carries enough privileged information to help the network learn to predict Q. Here, we employ the trajectory-based dataset generated with a joint variation $v = 1deg$ for the 7-DoF-7R robot, and $v = 1deg$ for the

[1] https://github.com/jacketdembys/iksolver

TABLE II: Experimental IK results obtained using the 100.000 samples from the test set of the 7-DoF Panda robot arm for all the investigated datasets. The orientation representation employed when training the networks is Roll-Pitch-Yaw. For the avg, min, and max values, the lower the better (↓) while for the percentage of error values less than $1(mm/deg)$, the higher the better (↑). The Learning-by-Example results are provided for $v = 1deg$.

Model	**Position** (*mm*)						**Orientation** (*deg*)					
	X		**Y**		**Z**		**Ro**		**Pi**		**Ya**	
	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑
Without Learning-by-Example												
MLP	82.2 (0.0/1025.2)	1.22%	89.4 (0.0/1071.3)	1.21%	69.2 (0.0/654.1)	1.57%	28.6 (0.0/179.8)	4.49%	16.3 (0.0/88.7)	4.91%	33.2 (0.0/179.9)	3.12%
RMLP	126.6 (0.0/1376.4)	0.62%	125.9 (0.0/1443.1)	0.62%	98.2 (0.0/856.8)	0.78%	45.5 (0.0/180.0)	1.83%	26.3 (0.0/89.7)	2.47%	56.2 (0.0/180.0)	1.34%
DMLP	117.8 (0.0/1318.0)	0.67%	117.5 (0.0/1405.4)	0.65%	92.8 (0.0/814.5)	0.81%	41.1 (0.0/180.0)	1.96%	25.8 (0.0/89.7)	2.55%	50.9 (0.0/180.0)	1.43%
With Learning-by-Example strategy ($v = 1deg$)												
MLP	0.7 (0.0/22.9)	77.29%	0.7 (0.0/20.1)	77.23%	0.5 (0.0/14.0)	85.44%	0.2 (0.0/3.0)	99.79%	0.2 (0.0/3.2)	99.78%	0.2 (0.0/3.7)	99.64%
RMLP	**0.1** (0.0/6.4)	**99.99%**	**0.1** (0.0/6.5)	**99.99%**	**0.0** (0.0/2.9)	**99.99%**	0.0 (0.0/0.8)	100%	0.0 (0.0/1.0)	100%	0.0 (0.0/1.5)	99.99%
DMLP	0.3 (0.0/5.7)	97.61%	0.3 (0.0/6.7)	97.62%	0.2 (0.0/3.2)	99.65%	**0.0** (0.0/0.8)	**100%**	**0.0** (0.0/0.9)	**100%**	**0.1** (0.0/1.2)	**99.99%**
Numerical Methods [25] ($v = 1deg$)												
SD [10]	0.2 (0.0/23.6)	99.99%	0.2 (0.0/15.8)	99.99%	0.1 (0.0/10.1)	99.99%	0.0 (0.0/1.8)	99.96%	0.0 (0.0/0.1)	100%	0.0 (0.0/0.1)	100%
SVF [9]	0.2 (0.0/1.0)	100%	0.2 (0.0/1.0)	100%	0.1 (0.0/0.7)	100%	0.0 (0.0/1.7)	99.96%	0.0 (0.0/0.1)	100%	0.0 (0.0/0.1)	100%
MX [27]	0.2 (0.0/1.0)	100%	0.2 (0.0/1.0)	100%	0.1 (0.0/1.0)	100%	0.0 (0.0/1.9)	99.93%	0.0 (0.0/0.8)	100%	0.0 (0.0/0.9)	100%

TABLE III: Experimental IK results obtained using the 100.000 samples from the test set of the 7-DoF GP66+1 robot arm for all the investigated datasets. The orientation representation employed when training the networks is Roll-Pitch-Yaw. For the avg, min, and max values, the lower the better (↓) while for the percentage of error values less than $1(mm/deg)$, the higher the better (↑). The Learning-by-Example results are provided for $v = 1deg$.

Model	**Position** (*mm*)						**Orientation** (*deg*)					
	X		**Y**		**Z**		**Ro**		**Pi**		**Ya**	
	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑	avg ↓ (min/max)	≤ 1 ↑
Without Learning-by-Example												
MLP	25.7 (0.0/542.7)	3.38%	25.6 (0.0/484.5)	3.35%	31.4 (0.0/695.6)	2.73%	4.2 (0.0/167.3)	17.59%	4.2 (0.0/79.0)	17.73%	4.9 (0.0/173.0)	15.44%
RMLP	36.4 (0.0/762.9)	2.20%	36.6 (0.0/773.5)	2.17%	42.6 (0.0/846.1)	1.80%	6.9 (0.0/175.1)	10.32%	6.6 (0.0/84.9)	10.64%	7.7 (0.0/178.5)	9.25%
DMLP	32.6 (0.0/612.1)	2.45%	32.5 (0.0/618.1)	2.44%	37.0 (0.0/757.4)	2.08%	6.3 (0.0/174.0)	11.08%	6.1 (0.0/83.0)	11.28%	7.1 (0.0/178.3)	10.00%
With Learning-by-Example ($v = 1deg$)												
MLP	0.7 (0.0/13.2)	77.88%	0.7 (0.0/13.6)	77.81%	0.7 (0.0/15.8)	73.90%	0.1 (0.0/1.4)	99.99%	0.1 (0.0/1.4)	99.99%	0.1 (0.0/1.7)	99.98%
RMLP	**0.1** (0.0/4.5)	**99.96%**	**0.1** (0.0/5.1)	**99.96%**	**0.1** (0.0/6.0)	**99.91%**	0.0 (0.0/0.5)	100%	0.0 (0.0/0.7)	100%	0.0 (0.0/0.6)	100%
DMLP	0.2 (0.0/3.2)	99.54%	0.2 (0.0/3.1)	99.54%	0.2 (0.0/4.0)	98.94%	**0.0** (0.0/0.4)	**100%**	**0.0** (0.0/0.5)	**100%**	**0.0** (0.0/0.5)	**100%**
Numerical Methods [25] ($v = 1deg$)												
SD [10]	0.1 (0.0/1.0)	100%	0.1 (0.0/1.0)	100%	0.1 (0.0/1.0)	100%	0.0 (0.0/1.8)	99.99%	0.0 (0.0/0.0)	100%	0.0 (0.0/0.1)	100%
SVF [9]	0.1 (0.0/0.5)	100%	0.1 (0.0/0.5)	100%	0.1 (0.0/0.3)	100%	0.0 (0.0/1.5)	99.99%	0.0 (0.0/0.0)	100%	0.0 (0.0/0.1)	100%
MX [27]	0.1 (0.0/1.0)	100%	0.1 (0.0/1.0)	100%	0.1 (0.0/1.0)	100%	0.0 (0.0/1.8)	99.91%	0.0 (0.0/1.0)	99.99%	0.0 (0.0/1.0)	99.99%

revolute joints, and $v = 1mm$ for the prismatic joint of the 7-DoF-2RP4R robot.

That is, we trained the baseline MLP and the proposed RMLP and DMLP networks for both robots with and without example pairs. Figures 5 and 6 show the average position (X, Y, Z in mm) and orientation (Ro, Pi, Ya in deg) errors for all the investigated networks against both input schemes; respectively for the 7-DoF-7R and 7R-2RP4R robots. We observe that Learning-by-Example strategy significantly improves the average position and orientation errors for all the investigated networks compared to the same networks trained without example pairs.

Tables II and III show the average (avg), minimum (min), and maximum (max) position (X, Y, Z) and orientation errors (Ro, Pi, Ya) errors for all the investigated models against both input schemes, respectively for the 7-DoF-7R and 7-DoF-2RP4R robot arms. For both robots, the RMLP- and DMLP-based DDIK solvers trained with the Learning-by-Example strategy outperformed the Plain MLP by predicting more IK solutions with less than $1mm$ average position and $1deg$ average orientation errors. The same Tables II and III also demonstrate that the proposed DDIK solvers achieve performance comparable to SD [10], SVF [9] and MX [27] numerical solvers.

For the 7-DoF-7R robot, RMLP achieved average position errors of $0.1mm$, $0.1mm$, $0.0mm$ respectively in X, Y, Z and

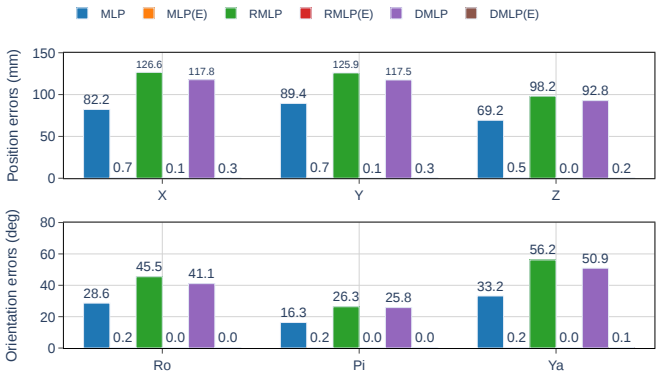


Fig. 5: Mean Absolute Position (X, Y, Z) and Orientation (Ro, Pi, Ya) Errors for the 7-DoF-7R Panda robot. The numbers on the bars represent the actual position errors for each model. The models labeled (E) denote the results obtained with the Learning-by-Example strategy following the input scheme illustrated in Figure 1.

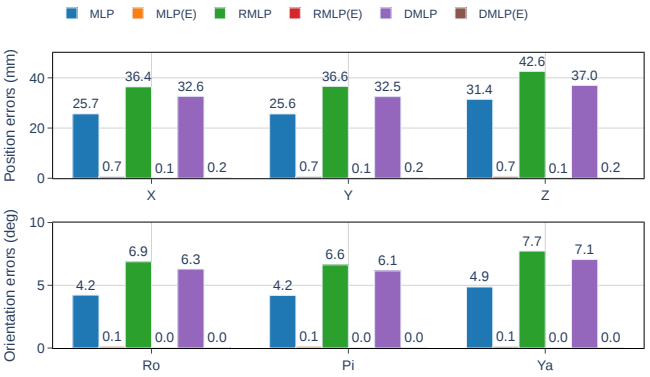


Fig. 6: Mean Absolute Position (X, Y, Z) and Orientation (Ro, Pi, Ya) Errors for the 7-DoF-7R GP66+1 robot. The numbers on the bars represent the actual position errors for each model. The models labeled (E) denote the results obtained with the Learning-by-Example strategy following the input scheme illustrated in Figure 1.

TABLE IV: Distribution of Mean Absolute Position (X, Y, Z) and Orientation (Ro, Pi, Ya) Errors for the 7-DoF Panda Robot within three error ranges: $\leq 1mm$, $(1,10]mm$, and $> 10mm$ for position errors and $\leq 1deg$, $(1,3]deg$, and $> 3deg$ for the orientation errors. All the values in the table are expressed in percentages (%) of the solutions falling within the specified error ranges. The Learning-by-Example results are provided for $v = 1deg$. For each error range, (↓) denotes the lower the percentage, the better the model while (↑) denotes the higher percentage, the better the model.

Error in	Error Range	Learning without Example			Learning-by-Example		
		MLP	RMLP	DMLP	MLP	RMLP	DMLP
X	≤ 1 ↑	1.22	0.62	0.67	77.29	**99.99**	97.61
	$(1,10]$ ↓	11.09	5.63	5.92	22.69	0.01	2.39
	> 10 ↓	87.68	93.74	93.41	0.02	0.00	0.00
Y	≤ 1 ↑	1.21	0.62	0.65	77.23	**99.99**	97.62
	$(1,10]$ ↓	10.84	5.58	6.00	22.75	0.01	2.38
	> 10 ↓	87.94	93.80	93.35	0.02	0.00	0.00
Z	≤ 1 ↑	1.57	0.78	0.81	85.44	**100.00**	99.65
	$(1,10]$ ↓	13.90	6.93	7.30	14.56	0.00	0.35
	> 10 ↓	84.54	92.29	91.88	0.01	0.00	0.00
Ro	≤ 1 ↑	4.49	1.83	1.96	99.79	**100.00**	**100.00**
	$(1,3]$ ↓	8.96	3.58	3.87	0.21	0.00	0.00
	> 3 ↓	86.56	94.59	94.17	0.00	0.00	0.00
Pi	≤ 1 ↑	4.91	2.47	2.55	99.78	**100.00**	**100.00**
	$(1,3]$ ↓	9.73	4.90	5.00	0.22	0.00	0.00
	> 3 ↓	85.37	92.64	92.45	0.00	0.00	0.00
Ya	≤ 1 ↑	3.12	1.34	1.43	99.64	**100.00**	**100.00**
	$(1,3]$ ↓	6.21	2.61	2.84	0.35	0.00	0.00
	> 3 ↓	90.67	96.05	95.73	0.01	0.00	0.00

TABLE V: Distribution of Mean Absolute Position (X, Y, Z) and Orientation (Ro, Pi, Ya) Errors for the 7-DoF GP66+1 Robot within three error ranges: $\leq 1mm$, $(1,10]mm$, and $> 10mm$ for position errors and $\leq 1deg$, $(1,3]deg$, and $> 3deg$ for the orientation errors. All the values in the table are expressed in percentages (%) of the solutions falling within the specified error ranges. The Learning-by-Example results are provided for $v = 1deg$. For each error range, (↓) denotes the lower the percentage, the better the model while (↑) denotes the higher percentage, the better the model.

Error in	Error Range	Learning without Example			Learning-by-Example		
		MLP	RMLP	DMLP	MLP	RMLP	DMLP
X	≤ 1 ↑	3.38	2.20	2.45	77.88	**99.96**	99.54
	$(1,10]$ ↓	27.75	19.18	21.09	22.11	0.04	0.46
	> 10 ↓	68.87	78.62	76.46	0.01	0.00	0.00
Y	≤ 1 ↑	3.35	2.17	2.44	77.81	**99.96**	99.54
	$(1,10]$ ↓	27.69	18.93	21.05	22.19	0.04	0.46
	> 10 ↓	68.97	78.90	76.51	0.01	0.00	0.00
Z	≤ 1 ↑	2.73	1.80	2.08	73.90	**99.91**	98.94
	$(1,10]$ ↓	22.63	15.97	18.10	26.08	0.09	1.06
	> 10 ↓	74.64	82.22	79.81	0.02	0.00	0.00
Ro	≤ 1 ↑	17.59	10.32	11.08	99.99	**100.00**	**100.00**
	$(1,3]$ ↓	30.86	19.71	21.14	0.01	0.00	0.00
	> 3 ↓	51.54	69.97	67.78	0.00	0.00	0.00
Pi	≤ 1 ↑	17.73	10.64	11.28	99.99	**100.00**	**100.00**
	$(1,3]$ ↓	30.95	20.21	21.49	0.01	0.00	0.00
	> 3 ↓	51.32	69.15	67.23	0.00	0.00	0.00
Ya	≤ 1 ↑	15.44	9.25	10.00	99.98	**100.00**	**100.00**
	$(1,3]$ ↓	27.88	17.92	19.29	0.02	0.00	0.00
	> 3 ↓	56.68	72.83	70.71	0.00	0.00	0.00

average orientation errors of $0.0deg$, $0.0deg$, $0.0deg$ respectively in Ro, Pi, Ya. For this same robot, the percentages of solutions obtained with RMLP within $1mm$ position error were 99.99%, 99.99%, 99.99% respectively for X, Y, Z; and within $1deg$ orientation error, 100%, 100%, and 99.99% respectively for Ro, Pi, Ya. While DMLP achieved average position errors of $0.3mm$, $0.3mm$, $0.2mm$ respectively in X, Y, Z and average orientation errors of $0.0deg$, $0.0deg$, $0.1deg$ respectively in Ro, Pi, Ya. For this same robot, the percentages of solutions obtained with DMLP within $1mm$ position error were 97.61%, 97.62%, 99.65% respectively for X, Y, Z; and within $1deg$ orientation error, 100%, 100%, and 99.99% respectively for Ro, Pi, Ya.

For the 7-DoF-2RP4R robot, RMLP achieved average position errors of $0.1mm$, $0.1mm$, $0.1mm$ respectively in X, Y, Z and average orientation errors of $0.0deg$, $0.0deg$, $0.0deg$ respectively in Ro, Pi, Ya. And, for this same robot, the percentages of solutions obtained with RMLP within $1mm$ position error were 99.96%, 99.96%, 99.91% respectively for X, Y, Z; and within $1deg$ orientation error, 100%, 100%, and 100% respectively for Ro, Pi, Ya. While DMLP achieved average position errors of $0.2mm$, $0.2mm$, $0.2mm$ respectively in X, Y, Z and average orientation errors of $0.0deg$, $0.0deg$, $0.0deg$ respectively in Ro, Pi, Ya. And, for this same robot, the percentages of solutions obtained with DMLP within $1mm$ position error were 99.54%, 99.54%, 98.94% respectively for X, Y, Z; and within $1deg$ orientation error, 100%, 100%, and 100% respectively for Ro, Pi, Ya.

For both robots, Tables IV and V show the distributions of errors within three error ranges expressed in terms of percentages of predicted IK solutions found to be less than $1mm$, between $1mm$ and $1cm$, and greater than $1cm$ for the position errors; and less than $1deg$, between $1deg$ and $3deg$, and greater than $3deg$ for the orientation errors. It can be observed that while MLP, RMLP, and DMLP trained with the Learning-by-Example strategy perform better than the same models trained without example pairs; RMLP and DMLP outperform MLP in terms of percentages of IK solutions

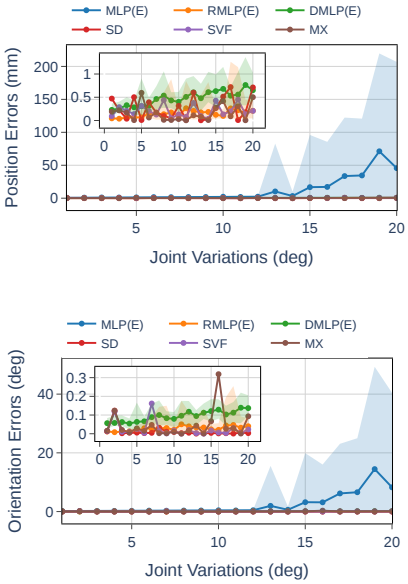


Fig. 7: Mean Absolute Position (**top plot**) and Orientation (**bottom plot**) Errors for the 7-DoF-7R Panda robot showing the impacts of joint variations v. The error bands represent the minimum and maximum values across 10 different runs of the MLP, RMLP, and DMLP models with diverse random initializations, while the solid dotted lines depict the average values of those runs. The models labeled (E) denote the results obtained with the Learning-by-Example strategy following the input scheme illustrated in Figure 1. The SVF [9] and MX [27] numerical solvers were run once by using the example joint as the initial configuration.

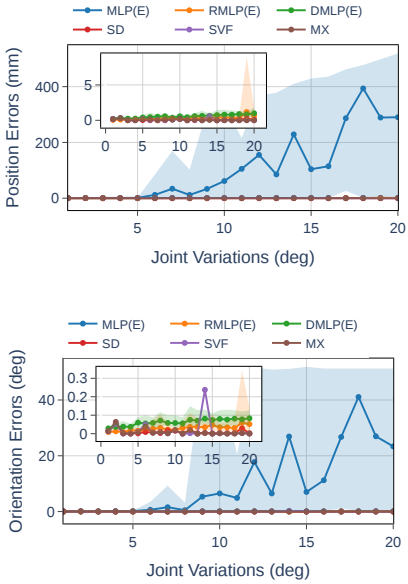


Fig. 8: Mean Absolute Position (**top plot**) and Orientation (**bottom plot**) Errors for the 7-DoF-2RP4R GP66+1 robot showing the impacts of joint variations v. The error bands represent the minimum and maximum values across 10 different runs of the MLP, RMLP, and DMLP models with diverse random initializations, while the solid dotted lines depict the average values of those runs. The models labeled (E) denote the results obtained with the Learning-by-Example strategy following the input scheme illustrated in Figure 1. The SVF [9] and MX [27] numerical solvers were run once by using the example joint as the initial configuration.

within the defined error ranges. From our observations, we found that residual and dense skip connections help in improving the performance of MLP-based DDIK solvers, particularly with the Learning-by-Example strategy. Overall, we note that RMLP outperforms DMLP by a small margin.

C. *Influence of the distance from the example pair*

To understand how far the example joint-pose pair needs to be from the query pose in the proposed Learning-by-Example strategy, we generated various datasets with increased joint variations v. These variations increase the distance between the pose in the example joint-pose pair and the query pose as illustrated in Figure 4. As mentioned before, we trained all the investigated networks for each of these datasets 10 times with different random initializations (random seeds). Figures 7 and 8 show minimum, average, and maximum pose errors as error bands based on the variations of v. We observed that as v increases, the IK predictions of the plain MLP-based DDIK solver get worse than those of RMLP and DMLP, hence showing the advantages of incorporating residual and dense skip connections. For the 7-DoF-7R robot, up to $v = 20deg$, RMLP (orange error band in Figure 7) and DMLP (green error band in Figure 7) exhibit average position and orientation errors respectively around $1mm$ and $0.2deg$; while around $v = 12deg$, the errors for MLP (blue error band in Figure 7) start getting larger. For the 7-DoF-2RR4P robot, up to $v = 20deg$ for the revolute joints and $v = 20mm$ for the prismatic joint, RMLP (orange error band in Figure 7) and DMLP (green error band in Figure 7) exhibit average position and orientation errors respectively around $2mm$ and $0.1deg$; while around $v = 5deg$, the errors for MLP (blue error band in Figure 8) start getting larger. For both robots, SD [10], SVF [9] and MX [27] numerical solvers were run once by using the example joint as the initial configuration and performed similarly to the proposed RMLP and DMLP DDIK solver as presented in Figures 7 and 8. Although the number of iterations for the numerical solvers increased in some cases as v grew larger, their performance remained consistent, as they continued iterating to achieve the desired pose error unless the maximum number of iterations was reached.

V. Conclusion

In this paper, we addressed the complex problem of Inverse Kinematics (IK) with a Data-Driven and Learning-by-Example framework, while investigating the utilization of residual and dense skip connections within an MLP-based learner for robotic manipulators. The resulting network architectures have demonstrated much improved results in approximating IK solutions of two redundant robotic manipulators: a 7-DoF-7R commensurate robot and a 7-DoF-2RP-4R incommensurate robot [25], [26]. For the Learning-by-Example cases, we introduced an input scheme that incorporates a joint-pose pair alongside the pose of interest, showcasing a simplified yet effective approach that achieves reliable IK solutions. Through a comprehensive performance comparison between the Plain MLP, ResNet-like MLP (RMLP), and DenseNet-like MLP (DMLP) models, we demonstrated that the skip connections-based networks performed better than the plain MLP when employing the

proposed example-based input scheme. On the other hand, we demonstrated that the models trained with the proposed Learning-by-Example strategy significantly outperformed the same networks trained without example pairs and achieved similar performance to Selectively Damped Least Squares (SD), Singular Value Filtering (SVF) and Mixed Inverse (MX) numerical solvers. Furthermore, we demonstrated that the example joint-pose pair does not need to be close to the example joint-pose of interest. That is, up to $v = 20deg$ for revolute joints and $v = 20mm$ for prismatic joints; RMLP and DMLP still provide reliable IK solutions. For future work, we plan to investigate collision avoidance and singularites when applying the proposed framework to redundant robotic manipulators. Furthermore, we also plan to leverage the proposed framework for multi-robots inverse kinematics learning with a single deep neural networks.

ACKNOWLEDGEMENT

Computational resources for this research have been supported by the NSF National Research Platform, as part of GP-ENGINE (award OAC #2322218).

REFERENCES

[1] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, 1955.

[2] J. Haviland and P. Corke, "A systematic approach to computing the manipulator jacobian and hessian using the elementary transform sequence," *arXiv preprint arXiv:2010.08696*, 2020.

[3] P. Corke and J. Haviland, "Not your grandmother's toolbox–the robotics toolbox reinvented for python," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 357–11 363.

[4] J. Demby's, Y. Gao, and G. N. DeSouza, "A study on solving the inverse kinematics of serial robots using artificial neural network and fuzzy neural network," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6.

[5] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," in *Computer Graphics Forum*, vol. 37, no. 6. Wiley Online Library, 2018, pp. 35–58.

[6] U. J. T. Demby's, "Use of jacobians for inverse kinematics of articulated robots: a study on approximate solutions," Ph.D. dissertation, University of Missouri–Columbia, 2020.

[7] R. Bensadoun, S. Gur, N. Blau, and L. Wolf, "Neural inverse kinematic," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1787–1797.

[8] F. L. Tagliani, N. Pellegrini, and F. Aggogeri, "Machine learning sequential methodology for robot inverse kinematic modelling," *Applied Sciences*, vol. 12, no. 19, p. 9417, 2022.

[9] A. Colomé and C. Torras, "Redundant inverse kinematics: Experimental comparative review and two enhancements," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5333–5340.

[10] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics tools*, vol. 10, no. 3, pp. 37–49, 2005.

[11] S. Tabandeh, C. Clark, and W. Melek, "A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering," in *2006 IEEE International Conference on Evolutionary Computation*. IEEE, 2006, pp. 1815–1822.

[12] T. J. Collinsm and W.-M. Shen, "Particle swarm optimization for high-dof inverse kinematics," in *2017 3rd international conference on control, automation and robotics (ICCAR)*. IEEE, 2017, pp. 1–6.

[13] F. Flacco, A. De Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 637–654, 2015.

[14] A. Liegeois *et al.*, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE transactions on systems, man, and cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[15] J. F. Dalmedico, M. Mendonça, L. B. de Souza, R. V. P. D. Barros, and I. R. Chrun, "Artificial neural networks applied in the solution of the inverse kinematics problem of a 3d manipulator arm," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–6.

[16] C. Acar and K. P. Tee, "Approximating constraint manifolds using generative models for sampling-based constrained motion planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8451–8457.

[17] B. Ames, J. Morgan, and G. Konidaris, "Ikflow: Generating diverse inverse kinematics solutions," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7177–7184, 2022.

[18] S. Farzan and G. N. DeSouza, "A parallel evolutionary solution for the inverse kinematics of generic robotic manipulators," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 358–365.

[19] S. Kim and J. Perez, "Learning reachable manifold and inverse mapping for a redundant robot manipulator," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4731–4737.

[20] F. Cursi, W. Bai, W. Li, E. M. Yeatman, and P. Kormushev, "Augmented neural network for full robot kinematic modelling in se (3)," *IEEE Robotics and Automation Letters*, 2022.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[23] C. Zhang, P. Benz, D. M. Argaw, S. Lee, J. Kim, F. Rameau, J.-C. Bazin, and I. S. Kweon, "Resnet or densenet? introducing dense shortcuts to resnet," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3550–3559.

[24] D. Xing, W. Xia, and B. Xu, "Kinematics learning of massive heterogeneous serial robots," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 535–10 541.

[25] J. Demby's, J. Uhlmann, and G. N. DeSouza, "Choosing the correct generalized inverse for the numerical solution of the inverse kinematics of incommensurate robotic manipulators," *arXiv preprint arXiv:2308.02954*, 2023.

[26] ——, "Achieving unit-consistent pseudo-inverse-based path-planning for redundant incommensurate robotic manipulators," *arXiv preprint arXiv:2308.02964*, 2023.

[27] J. Uhlmann, "A generalized matrix inverse that is consistent with respect to diagonal transformations," *SIAM Journal on Matrix Analysis and Applications*, vol. 39, no. 2, pp. 781–800, 2018.

[28] A. Guez and A. Ziauddin, "Solution to the inverse kinematics problem in robotics by neural networks," in *1988 IEEE International Conference on Neural Networks (ICNN)*. IEEE, 1988, pp. 617–624.

[29] A. Morell, M. Tarokh, and L. Acosta, "Inverse kinematics solutions for serial robots using support vector regression," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4203–4208.

[30] A. Volinski, Y. Zaidel, A. Shalumov, T. DeWolf, L. Supic, and E. E. Tsur, "Data-driven artificial and spiking neural networks for inverse kinematics in neurorobotics," *Patterns*, vol. 3, no. 1, 2022.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.