On Learning of Inverse Kinematics for Highly Redundant Robots with Neural Networks

Benedict Stephan, Ilja Dontsov, Steffen Müller, and Horst-Michael Gross

***Abstract*— The inverse kinematic problem for redundant robots is still difficult to solve. One approach is learning the inverse kinematic model with artificial neural networks, while the key challenge is the ambiguity of solutions. Due to the redundancy in the robot's degrees of freedom, there are multiple or even unlimited valid joint states bringing the end effector to a desired position. We show to what extent this problem influences the achievable accuracy of supervised training approaches depending on the number of degrees of freedom. To overcome the difficulties, a new training scheme is proposed, which uses the analytically solvable forward kinematics model. The new unsupervised training method uses random sampling in the joint state space and is not dependent on ambiguous tuples of joint values and end effector poses. We analyze the effect of the sample density on the remaining position error and show that additional soft constraints can easily be integrated in the training scheme, which offers the possibility to consider obstacle avoidance directly in the inverse kinematic model. Evaluations have been done using different robot models with up to 20 degrees of freedom, while not only position, but also the end effector's orientation at the goal point is considered.**

I. Introduction

Solving the inverse kinematic (IK) problem, which is to find the set of joint angles for a kinematic chain that brings the end effector in a desired pose in the Cartesian space, is a difficult task when the robot has more degrees of freedom (DoF) than necessary. While there are analytical solutions for simple robots, for redundant robots often only numerical iterative solutions (e.g. Damped Least Squares method [1]) are possible, which require additional constraints to select one among multiple possible solutions.

Already in the 1990's there were attempts to train neural networks to solve this problem, but mostly for low numbers of DoF [2, 3]. Usually, the models were trained using a set of captured tuples of end effector pose and respective joint state. These are either captured from a real robot, incorporating possible calibration errors, or it can be generated by means of a forward kinematic model, which is always solvable analytically. These approaches achieved good results when there is no ambiguity in the training data. This is ensured by either the low number of DoF or by introducing additional constraints for disambiguation. Even if a robot has more than six movable joints, the angular limits might be so tight that there are no problems with ambiguity. Demby [4] for example could achieve position errors in the range of mm with an MLP for standard robots with four to seven DoF, while the individual joint limits have been trimmed drastically.

This work has received funding from the Carl-Zeiss-Stiftung as part of the project engineering for smart manufacturing (E4SM).

Authors are with Neuroinformatics and Cognitive Robotics Lab, Technische Universität Ilmenau, 98693 Ilmenau, Germany.

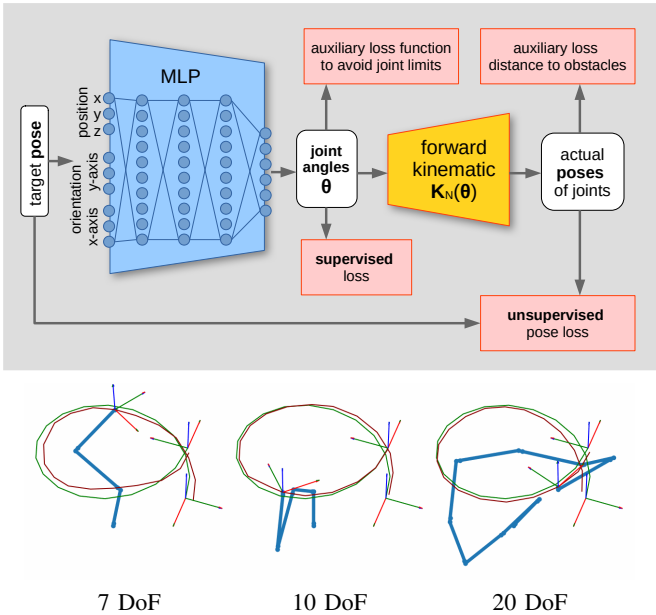

Fig. 1. Principle architecture of the proposed loss function for unsupervised training. Bottom row shows results of the end effector of increasingly complex robot models following a spiral trajectory. (green: target trajectory; red: actual trajectory; blue: the limbs of the robot arm)

Restricting the training dataset to an unambiguous subset of the configuration space seems to be counter-intuitive, since the capabilities of redundant robots to manage difficult situations with obstacles occluding some of the possible solutions unfortunately gets lost by such an approach.

In more recent publications, the training of networks for solving redundant robot geometries can be found. Bensadoun et al. [5] for example handle ambiguities explicitly by utilizing a sequence of networks each responsible for one joint in the chain. Each of them is considering the state of all the previous joints. So the individual models for higher level joints are trained with data that is conditioned on the state of the previous lower level joints.

Ensembles of networks have also been proposed to solve the problem by each specializing on a certain solution [6]. This ensemble approach can drastically reduce the remaining position error to a 10th.

In [7] the authors are discussing the problem of training networks for redundant robots. They propose a network that clusters the solutions and can query different solutions by using an additional input to the network for indexing. They

call this selective inverse kinematic (SIK). The same idea of selecting situation dependent sets of joint angles is found in some online controller applications. E.g., in [8] the current joint state of the robot is used as an additional input to the network. By means of selecting the closest among the possible ambiguous solutions, also higher order IK problems could be solved.

The supervised training of inverse kinematic models is also applied in the field of soft robotics, which tend to have a high degree of redundancy in their DoFs. In [9] they are learning from 10000 samples generated by exploration on real 9 DoF elephant trunk like robot, which is actuated pneumatically. In [10] this approach is analyzed theoretically.

When using machine learning models for solving a problem the question arises which architecture performs best. While most approaches use a simple Multi Layer Perceptron (MLP) as a baseline, in [11] different network architectures (LSTM, MLP, GRU) have been compared to an analytical model. Also unsupervised models like Kohonen Maps (KSOM) [12] have been used successfully for solving the IK for a highly redundant mobile manipulator having 14 DoF.

Since the MLP yields good results for the low DoF problems we also concentrate on this well-tried network for our own analysis. We think that the supervised training scheme is the main reason for the bad results reported for MLP solutions with high DoF robots and therefore concentrate on that most simple network architecture, knowing that the proposed training method might also improve more sophisticated networks.

A neural network model of the inverse kinematic usually is used in a controller for a robotic manipulator. Here, additional constraints like collision avoidance and avoiding of the joints' endpoints have to be considered. There also exist learned controllers for solving complete tasks by means of deep reinforcement learning. For example [13] learned a collision free, inverse kinematic model able to control a welding robot.

While many of the existing approaches either use a more complicated setup of multiple networks, constraint datasets or more sophisticated models such as recurrent neural networks, we intentionally keep our model simple. By changing the training method and the loss function to be optimized, we are also able to include additional capabilities to the IK solving neural network. For example, we can take the collision scene into account, enabling the network to select collision free configurations automatically during the training process. In sec. III-F we show an example for such additional benefits of our proposed training method.

Thus our key contributions are as follows:

- Introducing our approach of training inverse kinematic models in an unsupervised way to avoid contradictory training samples
- Analysis of the dependency of errors on model DoFs, joint limits, and sample distribution, while taking the orientation of the desired end effector pose into account
- Providing an example of integrating obstacle avoidance while solving the IK problem

II. Unsupervised inverse kinematics model training

The main problem with training neural networks for IK is the redundancy of joints. This results in multiple solutions for a given target pose and therefore yields contradictory training samples which would cause a neural network to average all provided solutions during supervised training to minimize the overall error. In Bensadoun 2022 [5] a comparative study of different neural network approaches for robots with 4, 6, and 7 DoF is shown. Here for the simple MLP an increase of the error from 123mm at 4 DoF to 591mm and 577mm for 6 and 7 DoF respectively could be observed, which proves that relationship quite well.

We propose to deal with these contradictions during training and not by filtering the dataset beforehand. By applying an unsupervised training regime we allow the neural network to converge on a single solution for a specific input pose while ignoring others. Instead of computing the training error directly over the predicted joint angles, we compute the resulting 6D pose of the end effector by using forward kinematics. Then we minimize the error between this pose and the target pose at the network's input. As computing forward kinematics can be formulated as simple matrix multiplications we can propagate the error to adapt the weights of our neural network through the forward kinematic model. An overview of our approach can be seen in Fig. 1. Our model is similar to an encoder decoder or autoencoder setup but with an analytical decoder. Although there may exist multiple samples during training that consist of different joint angles but result in the same 6d pose, our approach would allow the model to effectively decide on one of these solutions. This selection process takes place without handcrafted constraints or any additional information provided by a human.

Not needing to resolve contradictions during dataset generation, allows us to use random sampling in joint space. As we will show as part of our experiments, the remaining error of our trained models could be improved with more consideration during sampling, to achieve a more uniform coverage of the Cartesian space.

An additional advantage of using a neural network for solving the inverse kinematics problem is that one could easily integrate constraints to the learned solutions. For example one could compute the distance between joint positions and obstacles and add it to the error being minimized to achieve obstacle aware solutions. We showcase this possibility by using a static obstacle.

In contrast, a neural network trained in this manner does not allow for computing redundant solutions for a single target pose, as it focuses on a single solution during training. This makes our approach unsuitable for applications where multiple solutions are necessary, although a similar approach to [7] could allow conditioning the network with an extra input, allowing queries for multiple solutions.

A. Kinematic Model

For evaluating our approach we use an artificial robot arm which makes it easier to scale the number of joints. For

comparison and to rule out a possible bias in constructed kinematic chain we additionally provide results for the real robot model of a TIAGo[1] robot's arm which has 7 DoF.

The kinematic chain of our artificial robot model consists of two types of links which are used alternately. The first and all uneven numbered links have a revolute joint in x-direction followed by a 0.01m long limb. At the end of that, all even numbered joints are axial revolute joints with a length of only 0.15m. The corresponding transformations matrices for the forward kinematic model, which depend on the joint angles θ_i, are as follows:

$$\boldsymbol{A}_{2i}(\theta_{2i}) = \begin{bmatrix} cos(\theta_{2i}) & -sin(\theta_{2i}) & 0 & 0 \\ sin(\theta_{2i}) & cos(\theta_{2i}) & 0 & 0 \\ 0 & 0 & 1 & 0.15 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\boldsymbol{A}_{2i+1}(\theta_{2i+1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_{2i+1}) & -sin(\theta_{2i+1}) & 0 \\ 0 & sin(\theta_{2i+1}) & cos(\theta_{2i+1}) & 0.01 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

By means of these transformations, the forward kinematic model $\boldsymbol{K}_N(\boldsymbol{\theta})$ results from multiplication of all the joints:

$$\boldsymbol{K}_N(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{R}_N & \boldsymbol{t}_N \\ 0 & 1 \end{bmatrix} = \prod_{n=1}^{N} \boldsymbol{A}_n(\theta_n) \tag{3}$$

A given point $\hat{\boldsymbol{p}}_N$ in coordinates of the Nth joint (zero vector to get the actual joint's pose) can be transformed to world coordinates $\boldsymbol{p}_N$ by multiplying it with $K_N(\boldsymbol{\theta})$:

$$\boldsymbol{p}_N = \boldsymbol{K}_N(\boldsymbol{\theta})\hat{\boldsymbol{p}}_N \tag{4}$$

The forward model matrix $\boldsymbol{K}_N(\boldsymbol{\theta})$ also contains the description of the rotation $\boldsymbol{R}_N$ and the relative position $\boldsymbol{t}_N$ directly.

B. Data Generation

For training a model to solve the IK problem generating data is comparatively easy as one can simply sample randomly in the joint space within the robots limits and compute the joints' poses in Cartesian space by means of the forward kinematics. For our models we generate a test split by drawing 200 000 samples in this manner. As sampling the dataset is computationally simple, for training we randomly sample a new batch for each step, which is independent from the test split. This ensures that resulting errors are not caused by a general lack of training data. With a training split, which is functionally infinite, we train our models until it the loss converges. It is important to note that thereby it is possible for the model to be trained on samples which are similar or close to samples in the test set. For the problem of solving the inverse kinematics this is of low importance as we train our model on a fixed robot model. This removes the need for generalization while still being feasible for actual application.

By training our model with the unsupervised approach there is no need to account for possible contradictions in the datasets as all generated goal poses in Cartesian space are reachable by at least one joint configuration. Since we optimize only the difference of predictions and targets in Cartesian space, it makes no sense to measure errors in joint space as commonly seen in literature when the supervised training is used. Thus we concentrate on position error in our evaluation.

[1]TIAGo robot platform by PAL Robotics https://pal-robotics.com/

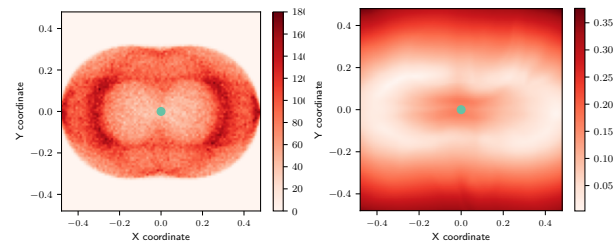

Fig. 2. Left: Cross section of the density of data points in Cartesian space resulting from uniform sampling 1M points in joint space of our 7 DoF artificial robot model; Cross section was taken with $z \in [0.1, 0.15]$. Right: corresponding L2 position error of the neural network IK model trained with the unsupervised method. The dot marks the origin of the robot arm.

The uniform distributed sampling in joint space unfortunately yields an uneven distribution of end effector poses in Cartesian space as shown in Fig. 2 (left). This might be a point for improvement, since regions with lower sample density usually correlate with higher errors. Tracking the sample density or applying some kind of hard sample mining in the sampling process could help to even out the sample density and therefore also reduce the variance of the position error.

C. Model Architecture

The network used for the evaluation (see Fig. 1) has the following structure: Inputs are the three dimensional goal coordinates of the end effector (x, y, z) normalized to a range of $[0, 1]$. For the experiments with orientation the desired rotation of the end effector is added as six additional inputs which represent two column vectors of the rotation matrix $\boldsymbol{R}_{in} = [\boldsymbol{a}_x \boldsymbol{a}_y \boldsymbol{a}_z]$. Since the third orthogonal vector $\boldsymbol{a}_z$ can be computed from these, the two direction vectors ($\boldsymbol{a}_x$ x-axis and $\boldsymbol{a}_y$ y-axis) are unique and represent the 3D version of a sine cosine representation of the three Euler angles.

Following a simple feed forward structure, the network has three hidden layers each using 100 neurons and a ReLU activation. In each hidden layer a batch normalization helps to stabilize the training.

The output of the network consists of neurons with linear activation function for the set of joint angles θ_i, which are normalized to match the joint's limits in the output range $[0, 1]$.

A second option for encoding the angles at the output called *biternion* is the representation by two neurons each. One encoding for the $sin(\theta_i)$ and the second encoding for the $cos(\theta_i)$. By means of the $arctan$ the actual θ_i can be reconstructed from these values. This representation is able to overcome the problems of periodical angle values which otherwise are highly discontinuous functions due to the wrap

around. Biternion encoding is a promising way for encoding angles at network outputs [14] and therefor was compared against the linear output. In our special case it is not expected that the biternions have a dramatic advantage over linear neurons, since our joints are not continuous and we do not have a wrap around point. (Details on the joints' limits are given in Section III-A.)

D. Loss Functions

For the *supervised* training which is used as a reference, the loss function is defined by the mean squared error (MSE) of the predicted θ_i to the ground truth θ_i^{gt}. In case of the biternion output the MSE is taken from $sin(\theta_i)$ and $cos(\theta_i)$ from the prediction to the $sin(\theta_i^{gt})$ resp. $cos(\theta_i^{gt})$ of the ground truth from the training data.

The loss function we propose for the *unsupervised* training does not make use of ground truth joint angles at all. As depicted in Fig.1, the predicted joint angles $\boldsymbol{\theta}$ are put into the forward kinematics $\boldsymbol{K}_N(\boldsymbol{\theta})$ which yields the individual coordinates of the Nth joint. Then the MSE of the goal point vector (network input) and the translation part $\boldsymbol{t}_N$ to be found in the third column of the $\boldsymbol{K}_N(\boldsymbol{\theta})$ yields the position error (L2 error). Since absolute errors in Cartesian space are less comparable for robots of different scales, we also provide the relative L2 error, which simply results from dividing by the robot arm's maximum range.

For the experiments that consider the orientation of the end effector, the first two columns of $\boldsymbol{K}_N(\boldsymbol{\theta})$ –neglecting the forth row– correspond to the two vectors $\boldsymbol{a}_x$ and $\boldsymbol{a}_y$ which define the target orientation at the network's input. These simply are compared by means of MSE as well yielding the rotation loss L_{rot}.

To get a more meaningful metric for the rotational difference we report the angle of the angle axis representation of the rotation $\boldsymbol{R}_{diff} = \boldsymbol{R}_{in}\boldsymbol{R}_N^{-1}$ with $\boldsymbol{R}_{in} = [\boldsymbol{a}_x\boldsymbol{a}_y\boldsymbol{a}_z]$ being the rotation of the target pose and $\boldsymbol{R}_N$ being the rotation of the predicted pose. This will be referred to as *rotation error* in the following experiments section.

As already claimed, it is easy to include additional loss terms further restricting the selection of the solution. E.g. in the last experiments we added a distance term L_{obs}, which sums up the truncated distances of the N individual joint positions (fourth columns of the $K_N(\theta)$) to a sphere defined in world coordinates (see Sec. III-F). This is exemplary for any more sophisticated representation of a static collision scene.

A further very important term is self collision avoidance. From the $K_N(\theta)$ it would easily be possible to compute mutual distances of the limbs of the robot. A violation of a distance threshold then could be added as an additional loss term restricting the selection to collision-free solutions.

A last loss term we propose is responsible for restricting the joint angles to their limits. This can directly be derived from the predicted joint angles as shown in Fig. 1.

E. Training Setup

For training our models we used the same hyperparameters for all experiments. We trained with a batch size of 100 and used the 1cycle scheduler with a maximum learning rate of 0.001 and Adam as optimizer. As described in Section II-B we randomly sampled each batch during training and trained until the model converged. Preliminary experiments showed convergence within 1 600 000 iteration. We evaluated the final weights on the test split and report errors as mean over all samples.

III. Discussion of Experimental Results

Using the described setup, we conducted four comparative experiments, the results of which are discussed in the following:

- To show that our artificial robot model is reasonable, we compare the results of the 7 DoF case with the real robot model of our TIAGo robot.
- A second series of trainings of the 7 DoF artificial model with varying angular limits shows that the limits actually influence the difficulty of the IK problem and therefor the achievable error.
- The main argument for our proposed unsupervised training method is a direct comparison of the supervised and unsupervised training for a series of increasingly complex robot models. Here the point where ambiguous solutions appear can be seen clearly. The supervised method fails completely while the unsupervised approach can handle even 20 dimensional joint spaces easily.
- As a last experiment we demonstrate the inclusion of static obstacles in the training yielding an inverse kinematic solver which considers collision avoidance.

A. Comparison of Artificial to Real Robot Model

We decided to conduct our experiments with an artificial robot model as described in Sec. II-A, which more clearly shows the problems induced by redundant DoF and the influence of limitations in the individual joints. This highly complex model yields position errors in the range of some cm, which is quite an unusable result for practical applications, but the robot's parameters have been chosen to be excessively difficult. To show that the general findings of the experiments are applicable to a real world robot as well, we used the supervised and unsupervised training for the TIAGo robot model, which has a 7 DoF arm. Table I summarizes the errors for different training setups on that model.

The TIAGo model reached an L2 error of 3.59cm using the unsupervised training and 5.9cm using the supervised approach. The supervised training fails to consider the target orientation and only reached 87.5° average error, while the unsupervised approach depending on the output encoding ends up in a range of 5° to 15°. These values are in a similar range for our artificial model with 7 DoF.

As a reference Köker et al. [6] reported position errors of 5.76–13.41mm for a 6 DoF robot arm using one feed forward neural network tuned for minimizing the error, which was not the primary goal for our experiments.

TABLE I: Results achieved on a 7 DoF real robot model for a TIAGo robot and for the artificial robot with 7 DoF using different output encoding and various loss functions for supervised and unsupervised training.

Model	Training	Output encoding	L2 pos. error [m]	rotation error [°]
Tiago	Sup.	Linear	0.0783	57.89
Tiago	Sup.	Biternion	0.0592	87.47
Tiago	Unsup. Pos	Linear	0.0130	115.40
Tiago	Unsup. Pos	Biternion	0.0122	120.02
Tiago	Unsup. Pos + Rot	Linear	0.0613	4.68
Tiago	Unsup. Pos + Rot	Biternion	0.0359	14.51
7 DoF Art.	Unsup. Pos + Rot	Linear	0.0889	3.75
7 DoF Art.	Unsup. Pos + Rot	Biternion	0.0445	4.63
7 DoF Art. (0.8 limits)	Unsup. Pos + Rot	Linear	0.0629	3.62

B. Restricting Joint Limits

In the literature the achievable error values for robots with the same DoF seem to have a high variance. According to our hypothesis, the difficulty and therefore the remaining errors are related to the degree of ambiguity in the dataset used for training. This ambiguity depends mainly on the DoF of the robot, but also the angular joint limits and the actual topology of the kinematic chain.

In a series of trainings, we incrementally reduced the angular limits of our 7 DoF artificial robot model. The maximum range of [-180°,180°] for even numbered joints and [-120°,120°] for the odd numbered joints has been scaled down to [-36°,36°] and [-24°,24°] respectively. The optimization considered position and orientation and used linear output encoding with the unsupervised training method.

The chart in Fig.3 shows the trend. For smaller limits the robot's operational area get more and more restricted and therefore also contradictory poses appear less often. For the restricted limits case the error values are in a practically usable range of 5mm and 1° which are comparable to the results of Demby [4] who trained supervised on an even more limited robot model.

C. Supervised vs. Unsupervised Training

The main focus of this paper is the comparison of the performance reached with the supervised and unsupervised training method for incrementally complex robots. We took the artificial robot model and extended the kinematic chain link by link while training the IK model with both methods.

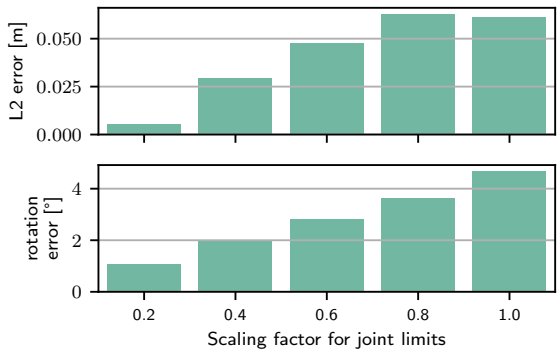

Fig. 3. Comparison of errors based on range of joint limits of 7 DoF artificial arm. Full limit's range of 1.0 means [-180°,180°] for the even numbered joints and [-120°,120°] for the odd joints.

Fig. 4 shows the results. According to the theory starting from 4 to 5 DoF the solutions become ambiguous and therefore the performance of the supervised training method significantly becomes worse. The chart shows also that the dramatic drop of performance is independent of the encoding of the joint angles at the network output, while the biternion version has a slightly better positional error.

The results of the unsupervised method seem counter-intuitive, since with higher DoF (over 8) the position and orientation errors drop again, having their maximum at 6 DoF resp. 8 DoF. This gives rise to the hypothesis that the unsupervised training process with redundant robots earlier finds a valid solution since the error landscape is more flat because there is a manifold of optimal solutions rather than only one. Obviously the method can disambiguate the possible solutions robustly. The unsupervised method finds a smooth continuous solution for the whole operational area. This is visible when following a smooth trajectory as shown in Fig. 1 (lower part).

Nevertheless, for the higher DoF problems the errors are worse than for the 4 DoF problem, which can relate to the network architecture limitations but also to the data sampling.

For 6 DoF the robot is not redundant for the position plus orientation problem and there is a finite set of algebraic solutions only. For the supervised method these already seem to cause the network to average the solutions which causes the dramatic rise of errors at 4 and 5 DoF for orientation and position respectively.

The experiment also showed that the difference between supervised and our training method is present for the biternion encoding of the joint angle outputs. This encoding has slightly better position errors, but the orientation error increases. There seems to be a trade-off, while the overall error is limited. The classical supervised training independent

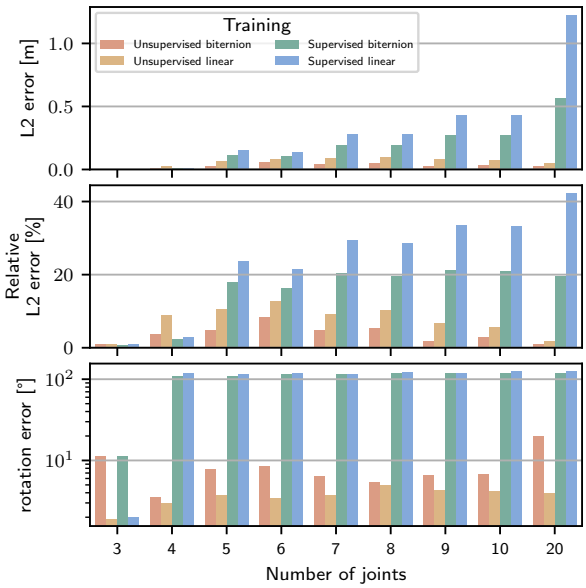

Fig. 4. Comparison of L2 and rotation error dependend on number of DoFs for supervised and unsupervised training on our artificial robot model. The two encodings (linear and biternion) for the joint angles also have been tested separately.

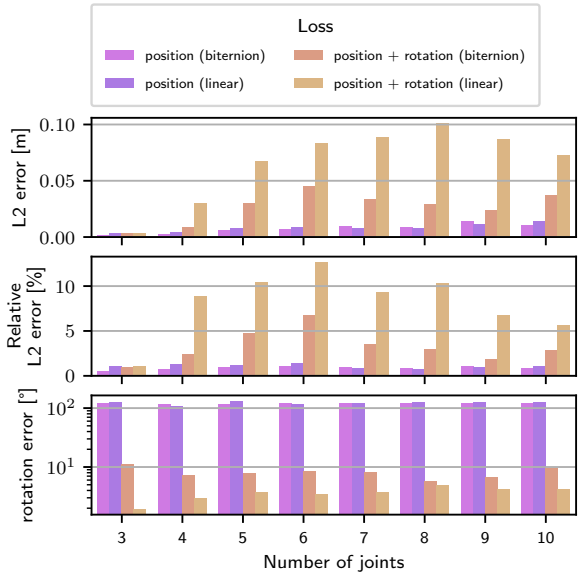

Fig. 5. Comparison of unsupervised training with position only and position plus orientation as target. The output encodings (linear vs. biternion) also have been varied.

of the encoding is not suitable to learn IK with a MLP for redundant robots without additional constraints on the dataset.

D. Position-only Training

In many publication only the position error is evaluated neglecting the orientation at the target point. Especially in early publications for low DoF (2 to 4) networks have been trained to reach a position regardless of the orientation of the end effector.

Whereas supervised training implicitly learns the orientation of the end-effector, unsupervised training must explicitly ensure that this is the case by introducing respective loss terms (see Sec. II-D). We also tried to train the unsupervised model only on the position loss, which yields the results given in Fig. 5.

The general trend, as discussed above, can also be found in this chart, but there is a clear benefit for the position only training. The positional error for all DoF configurations is reduced to less than half the value of the position and orientation target. Of course, the rotational errors far exceed the results for the position only case, since the network does not receive an appropriate loss signal.

It is remarkable, that even with 20 DoF the unsupervised training method reached a relative position error that is not bigger than that of a 3 DoF model. This makes the approach interesting for the soft robotics domain, where robots with highly redundant topology may exist.

E. Distribution of the Error in the Workspace

The errors reported so far seem to be very high compared to practical implementations found in literature. Therefore, we searched for a possible explanation and analyzed the distribution of the position error over the operational area of the robot. In many publications the error along special testing trajectories is reported, where the trajectories are in a good-natured, limited subspace of the reachable area. This also improves the error for our experiments. When concentrating on the spiral test trajectories shown in the Fig.1 the errors reduce almost to a half compared to the uniform distributed test samples. See Table II for exact numbers.

TABLE II: Average L2 position error for different DoF models along a test trajectory vs. measured with a uniformly distributed test set.

DoF	trajectory	uniform
7	0.0248	0.0293
10	0.0139	0.0304
20	0.0188	0.0337

Dependent on our testing method with randomly sampled poses, regions which are difficult to reach are present causing a higher average error. Fig. 2 on the right side shows a cross section of the Cartesian operational space of the 7 DoF robot at a z-position of 10 to 15 cm. In this plane the average position error is color coded. One can clearly see that the errors linearly grow outside the range of the robot arm, but already increases within the sampled space. Also in the center close to the origin of the robot arm there seem to be poses that are harder to reach than in the mid range areas. The density of data samples, which is shown on the left side of Fig. 2 seems to be a good explanation for the error distribution. The regions of higher error correspond to regions with lower sample density. This is consistent with the theoretical behavior of a network training which tries to minimize the average error over the whole dataset. Individual harder to reach samples in a low density region participate only little to the overall error and the network concentrates on minimizing the error in more densely sampled regions.

These insights provide an impetus to rethink the sampling process. In order to make the error distribution more homogeneous, one possible solution would be to make the distribution of training samples more uniform in the Cartesian space. In future work we want to work on that problem by creating a kind of bootstrapping during training. Samples could be approximately uniformly sampled, utilizing the partly trained IK solving network itself. Other options are a tracking of the sample density in Cartesian space and re-weighting the newly drawn samples by the inverse of the density. Also a gradually changing sample set in the training batch is possible while the evaluation of the error for the batch samples helps to identify harder examples that might be reused in the next batch together with new random samples.

F. Obstacle Avoidance

To showcase how our approach can be easily extended to incorporate constraints or more dynamic scenarios, we perform a small experiment to enable collision avoidance for our trained models. As a proof of concept, we added the distance of the joint positions to a spherical obstacle as an additional loss function during training. The obstacle loss L_{obs} results from the $\boldsymbol{t}_N$ from the $\boldsymbol{K}_N(\boldsymbol{\theta})$ and the center $\boldsymbol{c}$ of the sphere with the radius r as:

$$L_{obs} = ReLU\left(-min_N\left(|\boldsymbol{t}_N - \boldsymbol{c}| - r - 0.1\right)\right) \quad (5)$$

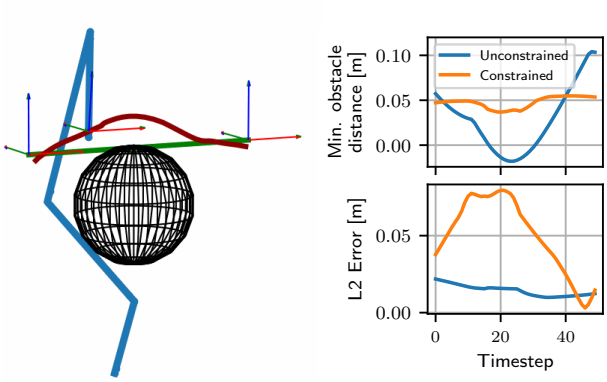

Fig. 6. Example results of 10 DoF arm (blue line) with a sphere as static obstacle; Green: target trajectory; Red: actual trajectory by IK network; On the right the minimum obstacle distance to all joints of the robot and the actual L2 error to the target trajectory are given over time. The training without additional loss functions (blue graph) is compared to the model with obstacle distance loss L_{obs} (orange graph).

The 0.1m here is a safety margin, the robot should keep to the obstacle when possible.

The resulting behavior of the IK model is exemplary shown in Fig. 6. The comparison of the minimum joint to obstacle distance over the pass along the green trajectory (see right side of the figure) shows, that the extended model finds a trade-off between the distance to the target trajectory and the distance to the obstacle inside the safety margin and avoids the obstacle completely otherwise. Therefore, the L2 error (orange curve lower diagram) is increased in the region where the sphere comes close to the target trajectory. The model without the L_{obs} term (blue plots) completely ignores the sphere causing collisions even if the actual trajectory is outside the sphere.

This behavior demonstrates the claimed ability to consider collision avoidance in static scenes during the training of the redundant IK model. In order to extend the obstacle awareness to dynamic objects, the network needs an additional input describing the current scene. As there are multiple approaches to represent the environment of a robot, e.g. though point clouds or voxel based approaches, which can also be used as an input to a neural network (e.g. PointNet[15]), it would be promising to train a network capable of handling dynamic environments in future.

IV. Conclusion and Future Work

We could show that the lack of performance of simple feed forward neural networks (MLPs) for the inverse kinematic task is related to the contradictory data samples when using supervised training. The proposed unsupervised training proved to be insensitive to ambiguity and even performed better for highly redundant robots. We also showed that learning to solve the orientation of a robot's end effector is possible but harder than using only a target position.

An analysis of the distribution of the remaining errors showed a correlation to the sample density. Therefore, in future work we will concentrate on improving the sampling strategy. Also the ability to consider additional constraints gives rise to further investigations. We plan to train obstacle sensitive inverse kinematic models by adding an abstract encoding of a dynamic collision scene at the network input while using the distance of the robots limbs to the scene as additional loss during training. Despite the remaining error and the lack of redundant solutions for a single target pose, the IK network with obstacle avoidance can therefore be helpful as a fast sampler for motion planning approaches like [16]. By means of such a model, the planning time in cluttered environments might be drastically reduced, as it can provide an initial guess for the required joint configuration. While this initial guess might not reach the target pose exactly, subsequent optimization steps can reduce the remaining error.

References

[1] X. Wang, X. Liu, L. Chen, and H. Hu, "Deep-learning damped least squares method for inverse kinematics of redundant robots," *Measurement*, vol. 171, p. 108821, 2021.

[2] B. B. Choi, *Inverse kinematics problem in robotics using neural networks*. Lewis Research Center, 1992, vol. 105869.

[3] S. Tejomurtula and S. Kak, "Inverse kinematics in robotics using neural networks," *Information sciences*, vol. 116, 1999.

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

[6] R. Köker, T. Çakar, and Y. Sari, "A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators," *Engineering with Computers*, vol. 30, pp. 641–649, 2014.

[7] C.-K. Ho and C.-T. King, "Automating the learning of inverse kinematics for robotic arms with redundant dofs," 2022. [Online]. Available: https://arxiv.org/abs/2202.07869

[8] H. Toshani and M. Farrokhi, "Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: a lyapunov-based approach," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 766–781, 2014.

[9] M. Rolf and J. J. Steil, "Efficient exploratory learning of inverse kinematics on a bionic elephant trunk," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 6, pp. 1147–1160, 2013.

[10] T. G. Thuruthel, E. Falotico, M. Cianchetti, and C. Laschi, "Learning global inverse kinematics solutions for a continuum robot," in *ROMANSY 21-Robot Design, Dynamics and Control: Proceedings of the 21st CISM-IFToMM Symposium, Udine, Italy*. Springer, 2016.

[11] J. S. Toquica, P. S. Oliveira, W. S. Souza, J. M. S. Motta, and D. L. Borges, "An analytical and a deep learning model for solving the inverse kinematic problem of an industrial parallel robot," *Computers & Industrial Engineering*, vol. 151, 2021.

[12] R. Raja, A. Dutta, and B. Dasgupta, "Learning framework for inverse kinematics of a highly redundant mobile manipulator," *Robotics and Autonomous Systems*, vol. 120, p. 103245, 2019.

[13] J. Zhong, T. Wang, and L. Cheng, "Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics," *Complex & Intelligent Systems*, 2021.

[14] B. Lewandowski, D. Seichter, T. Wengefeld, L. Pfennig, H. Drumm, and H.-M. Gross, "Deep orientation: Fast and robust upper body orientation estimation for mobile robotic applications," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 441–448.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[16] St. Mueller, B. Stephan, and H.-M. Gross, "MDP-based motion planning for grasping in dynamic szenarios," in *Europ. Conf. on Mobile Robotics (ECMR), Bonn, Germany*. IEEE, 2021, p. 8 pages.