Neural Networks for Inverse Kinematics Problems in Robotics

Franziska Schwaiger
Matriculation number: 03658670

Thomas Barthel Brunner
Matriculation number: 03675118

Introduction

In our project, we are evaluating the feasibility of using neural networks for inverse kinematics problems in robotics. In this report, we outline the progress made and discuss the results we have obtained.

Methods

Robot Simulations

To train and test our models, we developed simulations of planar robotic arms with revolute joints. As we are interested in testing the limits of our models, we created simulations of arbitraty degrees of freedom (DoFs). In general, the forward kinematics equations of a planar robot arm with link lengths l_i and N revolute joints with joint angles θ_i can be described as:

$$x_{\text{TCP}} = \sum_{i=1}^{N} l_i \cos\left(\sum_{j=1}^{i} \theta_j\right),\ y_{\text{TCP}} = \sum_{i=1}^{N} l_i \sin\left(\sum_{j=1}^{i} \theta_j\right) \tag{1}$$

These equations were then modified for fast vectorized numerical computation. The resulting operation and be seen in Equation 2, with joint angles vector $\boldsymbol{\theta} = [\theta_1, \theta_2, ...]^{\text{T}}$, the upper triangular matrix with ones as elements $\boldsymbol{U}$ and the link lengths vector $\boldsymbol{l} = [l_1, l_2, ...]^{\text{T}}$.

$$x_{\text{TCP}} = \cos\left(\boldsymbol{\theta}\ \boldsymbol{U}\right)\boldsymbol{l},\ y_{\text{TCP}} = \sin\left(\boldsymbol{\theta}\ \boldsymbol{U}\right)\boldsymbol{l} \tag{2}$$

In our current setup, we do not take the angle of the tool center point (TCP) into consideration. Thus, the location of the TCP is defined by its x, y coordinates. As a result of this, all simulations with more than 2 DoF can have up to infinite solutions of the inverse kinematics for a given TCP position. The 2 DoF robot arm can have up to two solutions.

Dataset

The dataset used for training is composed of 10^6 samples of robot configurations for each DoF. In our experiments, we used simulations with 4, 6, 8 and 10 DoF. Fig.1 shows an illustration of the datasets. Each configuration was sampled from a normal distribution $\theta_i \sim \mathcal{N}(\mu = 0, \sigma = 0.2)$. Thus, the dataset is composed mostly of configurations in which the robot arm is extended. This reflects real-world use cases of robot arms, which have limited workspaces and whose tasks are focused on one section of the workspace. Moreover, limiting the range of the joints improved the performance of the networks, as this avoids the discontinuity in the angles at $\theta = \pi$.

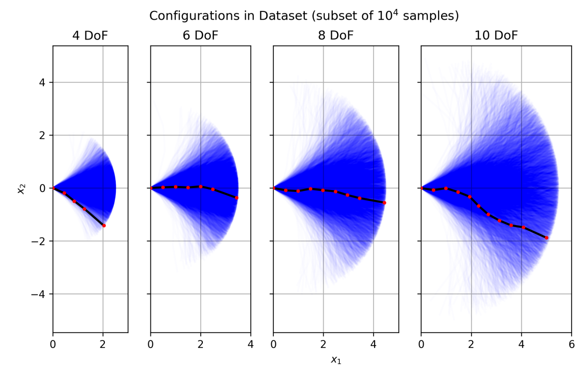


Fig. 1. Subset of datasets used for training and testing.

Considering Singularities

As already mentioned in the previous section, we avoid discontinuities by restricting the configuration space of the robot. Yet we implemented two approaches which consider this discontinuity in the angles at $\theta \in (-\pi, \pi)$ to extend the implemented networks to inverse kinematics problems with general configuration spaces.

The first approach is a non-minimal parameterization of the configuration space as it uses a vector-based representation which was already described in the midterm report to avoid singularities at the boundaries of the joint angles. The drawback of this approach is that the input space is doubled which becomes more and more a problem when increasing the amount of revolute joints of the planar robot.

Therefore, we also considered an alternative approach where we just consider the joint angles resulting in a minimal-parameterization of the configuration space, again. The discontinuities are now considered in the loss function for the reconstruction error of the joint angle space. It ensures that, for example when having a ground truth joint angle $\theta_{gt} = 178°$ and a predicted joint angle $\theta_{pred} = -178°$, the difference between these two angles is only $4°$ instead of $356°$ resulting in a smaller loss.

Network Architectures

As described in the midterm report, we implemented two network architectures for generating the full posterior distribution of the joint angles: Unlike Variational Autoencoders (VAE), conditional Variational Autoencoders (cVAE) are able to control the data generation process by conditioning the latent space z. Considering the application of inverse kinematics, samples are drawn from $p(z) \sim N(0,1)$ and the predicted posterior distribution of the joint angles is then generated conditioned on the end-effector position.

Invertible Neural Networks (INN) model a bijective mapping from the joint angles to the end-effector position by stacking invertible blocks together. Here, the dimensionality of the joint angles x is the same as the concatenated output consisting of the latent space z and the end-effector position y. The predicted posterior of the joint angles is then generated similar to cVAE by sampling from $p(z) \sim N(0,1)$ and then running the network backwards conditioned on y.

Additionally, INNs share properties with normalizing flows [1], [2] which gradually transform a normal density into the desired data density. They are also bijective and for both normalizing flows and INNs, a tractable Jacobian exists which allows explicit computation of posterior probabilities. As the focus in this work lies on the comparison between cVAE and INN, normalizing flows are not considered anymore.

As stated in [3], the INN only needs to be trained on the well-understood forward process and then the inverse process can be obtained for free by just running the network backwards at prediction time. The authors also mention that results can be improved with additional unsupervised backward training. When applying INNs to the inverse kinematics problem, we found that to match the performance of the cVAE, unsupervised backward training is crucial. This results in two network passes per training step to accumulate the gradients and slows down training a lot. Additionally, when training INNs the parameters need to be adjusted carefully as training can become unstable.

Hyperparameter Optimization

For our tests, we trained a network for each robot simulation (DoF). Thus, the sizes and selection of hyperparameters of each network had to match the complexity of the simulation. To compare the performance of the networks across different DoFs in a fair setting, we used Tune [4] and Scikit Optimize [**?**] to perform a black-box optimization of the hyperparameters. For each model, 100 sets of hyperparameters were sampled and were then trained using a subset of the main dataset (10^4 samples), which helped reduce the computation time. The ranges of hyperparameters used can be seen in Table I and II. These values were based on intuition and previous experience in training these networks.

Implementation

The previous two model architectures (cVAE and INN) have been implemented in PyTorch and are inspired by existing implementations in [5], [6]. We utilized the Google Compute Engine to train our models.

EXPERIMENTAL EVALUATION

Evaluation protocol

We extended the DoF of the robots up to 10 DoF and performed hyperparameter optimization (HPO) for the hyperparameters for DoF $= [4, 6, 8, 10]$. Due to HPO, the number of trainable parameters differs between cVAE and INN. The batch size is kept fixed to 1000 for both networks. For the cVAE, we optimized the learning rate, the weight decay, the number of hidden layers for the encoder and decoder and the amount of neurons per layer. For the INN, we optimized the learning rate, the weight decay, the number of coupling layers, the amount of layers per subnet and also the number of neurons per subnet layer. Both networks have been trained for 60 epochs on a dataset with 1 million samples and a train and test split of 70 and 30, respectively. For evaluation, we used again the two evaluation metrics which have been described in the midterm report:

1. The average posterior mismatch between the distribution generated by the model and the ground truth estimate obtained via rejection sampling. For generating the posterior, we used 100 samples per ground truth position and averaged over 1000 samples from the test dataset.

2. The average re-simulation error as the mean squared distance between the ground truth end-effector position and the re-simulated end-effector position obtained from the predicted joint angles. The re-simulation error was averaged over the whole test dataset.

Results

The hyperparameters for the cVAE and INN chosen by HPO are depicted in Table III and IV, respectively. Because of HPO, the number of parameters per robot is not approximately equal, anymore. As it can be seen in the tables, the best architectures are not the architectures with the most numbers of trainable parameters. This also shows that the hyperparameters search space is sufficient.

For both networks, the mismatch of the posterior distribution and the re-simulation error is plotted over the number of parameters in Fig. 2 and 3, respectively. Each plotted point is labelled with the corresponding robot of different DoF. As it can be seen, when increasing the number of DoF, the number of parameters does not have to be increased, as well. This is a result of the HPO. The mismatch of the predicted posterior (see Fig. 2) with the ground truth posterior obtained from rejection sampling is less for the INN as for the cVAE. But in contrast to that, the cVAE performs consistently better on the re-simulation error as the INN (see Fig. 3).

In Fig. 5a and 5b, this behaviour is described in more detail: A convex hull is drawn around the 97th percentile of the re-simulated points (green dots) and the corresponding ground truth end-effector position (red dot) at $(x, y) = [3.47, 0.19]$ of a 10 DoF robot for the cVAE and INN, respectively. As it can be seen, the area of this convex hull is larger for the INN (0.85) than for the cVAE (0.15). But despite of having a larger re-simulation error, the INN generates the full posterior distribution of the joint angles much better than the cVAE, as it can be seen in Fig. 7.

CONCLUSION

In this work, we evaluated through simulation experiments the feasability of using neural network architectures to fully

TABLE I
RANGES OF INN HYPERPARAMETERS USED DURING HYPERPARAMETER SEARCH.

	Learning rate	# Subnet layers	# Coupling layers	# Neurons per layer	Weight decay
Min	0.0001	3	6	100	0.00001
Max	0.005	7	10	300	0.001

TABLE II
RANGES OF CVAE HYPERPARAMETERS USED DURING HYPERPARAMETER SEARCH.

	Learning rate	# Layers	# Neurons per layer	Weight decay
Min	0.0001	3	200	0.00001
Max	0.01	15	500	0.001

TABLE III
HYPERPARAMETERS FOR CVAE TRAINED ON A PLANAR ROBOT WITH REVOLUTE JOINTS

DoF	Learning rate	# Layers	# Neurons per layer	Weight decay	# Trainable parameters
4	0.009	3	230	0.0002	217128
6	0.0041	3	200	0.00001	166814
8	0.0001	3	500	0.00001	1022020
10	0.00063	5	400	0.00029	1303226

TABLE IV
HYPERPARAMETERS FOR INN TRAINED ON A PLANAR ROBOT WITH REVOLUTE JOINTS

DoF	Learning rate	# Subnet layers	# Coupling layers	# Neurons per layer	Weight decay	# Trainable parameters
4	0.0009	5	9	100	0.00008	1108872
6	0.001	5	7	180	0.0003	2772084
8	0.0014	4	9	115	0.0004	997884
10	0.002	5	7	170	0.0005	2494380

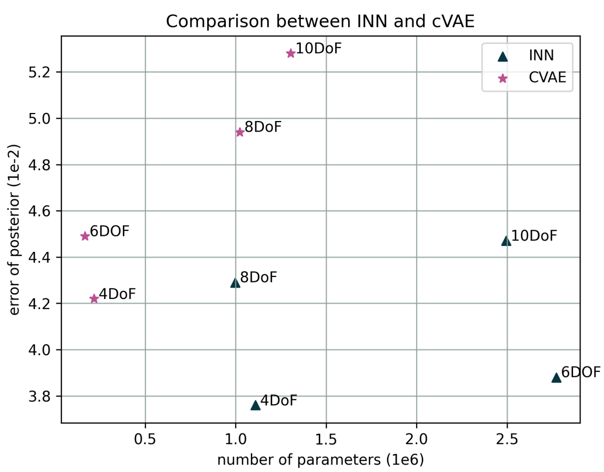


Fig. 2. Average posterior mismatch between the distribution generated by the model and the ground truth estimated obtained via rejection sampling.

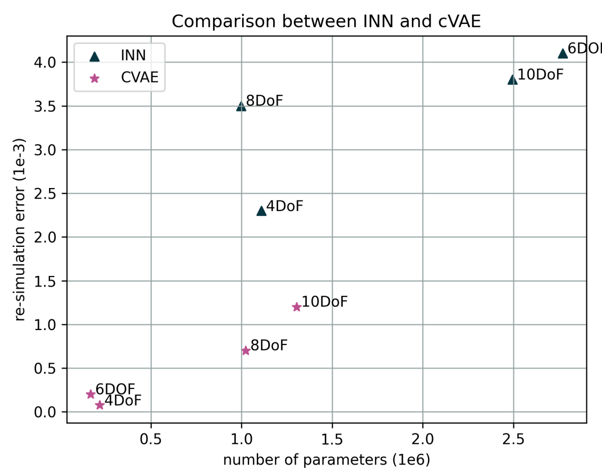
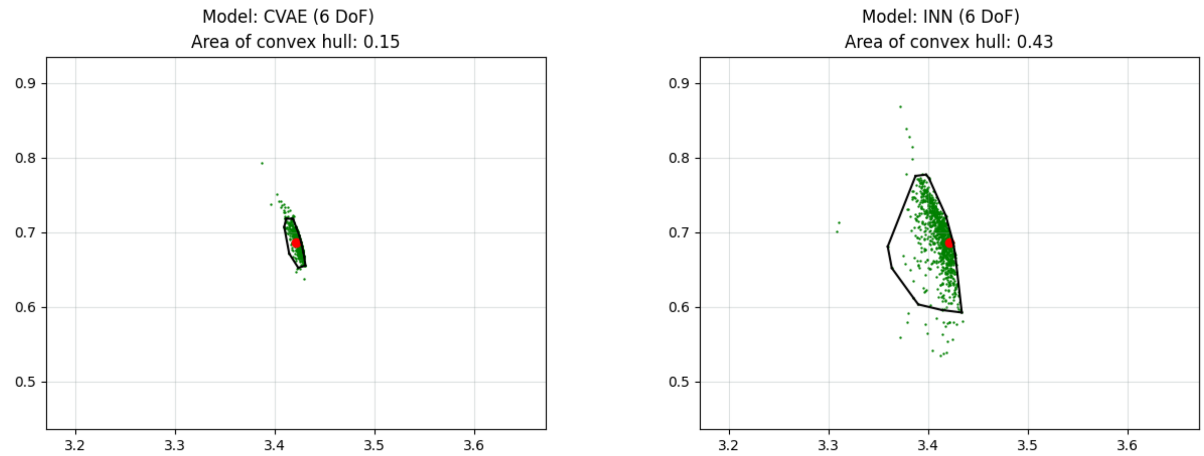


Fig. 3. Average re-simulation error as the mean squared distance between ground truth end-effector position and the re-simulated end-effector position obtain from the predicted joint angles.

generate the posterior distribution of the hidden parameter space in the context of inverse kinematics problems in robotics. For this, we studied invertible neural networks and compared them to conditional Variational Autoencoders using as a baseline. At first, we constructed simple planar robots with up to 4 revolute joints and then extended the simulations with more complex planar robots with up to 10 revolute joints. To find the best set of hyper parameters we performed Bayesian optimization.

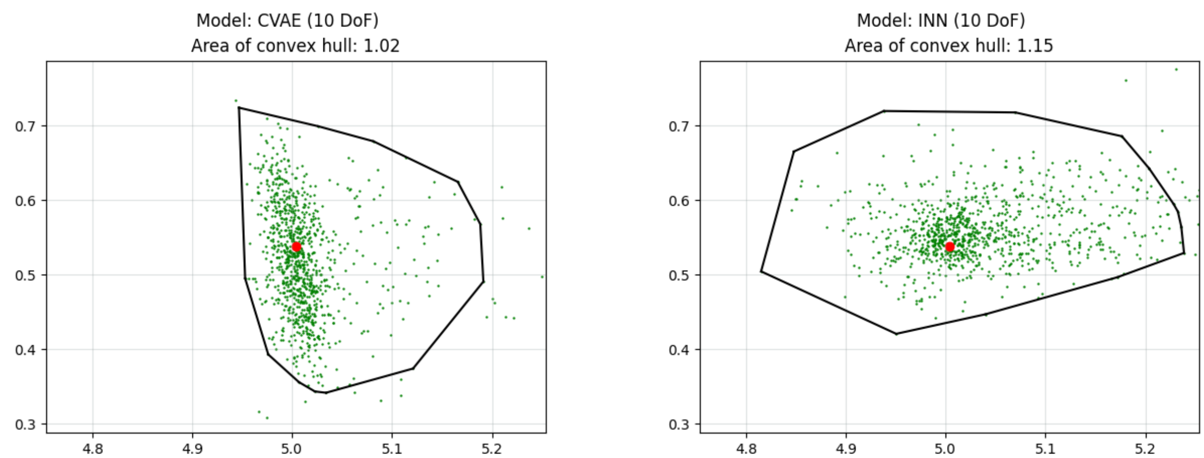
Using evaluation metrics defined in previous research, given the end-effector position we found that cVAEs are better in predicting the values of the joint angles which, when applied to forward kinematics again, result in being closer to the ground truth end-effector position. In contrast, experiments show that INNs better recover the full posterior distribution over the joint angle space.



(a) cVAE

(b) INN

Fig. 4. Area of the convex hull of the 97th percentile of the re-simulated end-effector coordinates with the ground truth end-effector position at $(x, y) = [3.42, 0.68]$ and 1000 samples. Number of DoF: 6.



(a) cVAE

(b) INN

Fig. 5. Area of the convex hull of the 97th percentile of the re-simulated end-effector coordinates with the ground truth end-effector position at $(x, y) = [5.00, 0.53]$ and 1000 samples. Number of DoF: 10.

Unlike stated in previous research, we had to perform an additional unsupervised backward training to match the performance of the cVAE resulting in slowing down the training a lot. Also, INNs tend to be more unstable in the training process as cVAEs. Despite this, INNs are an interesting and powerful tool in not only recovering the full posterior parameters distribution but also being able to explicitly compute the posterior probability by using tractably Jacobians.

References

[1] E. G. Tabak and E. Vanden-Eijnden, "Density estimation by dual ascent of the log-likelihood," *Communications in Mathematical Sciences*, vol. 8, no. 1, pp. 217 – 233, 2010. [Online]. Available: https://doi.org/

[2] E. Tabak and C. Turner, "A family of nonparametric density estimation algorithms," *Communications on Pure and Applied Mathematics*, vol. 66, no. 2, pp. 145–164, Feb. 2013, copyright: Copyright 2012 Elsevier B.V., All rights reserved.

[3] L. Ardizzone, J. Kruse, S. J. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe, "Analyzing inverse problems with invertible neural networks," *CoRR*, vol. abs/1808.04730, 2018. [Online]. Available: http://arxiv.org/abs/1808.04730

[4] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

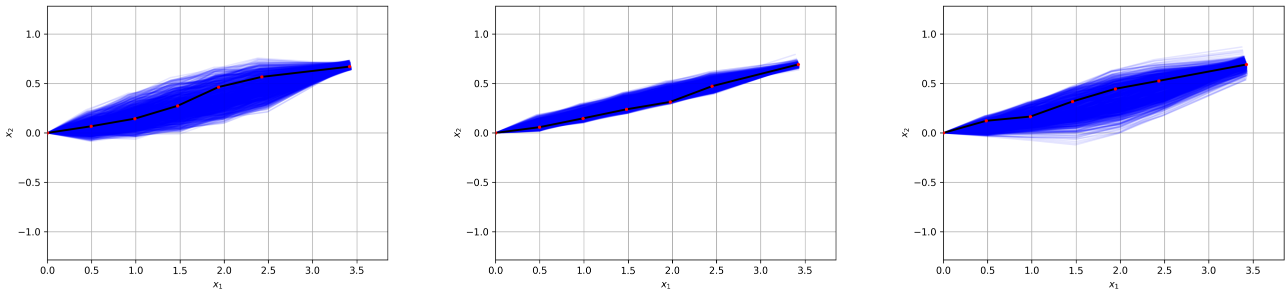
[5] graviraja, "pytorch-sample-codes," https://github.com/graviraja/pytorch-sample-codes, 2019.

[6] "Freia," https://github.com/VLL-HD/FrEIA/blob/master/FrEIA/, 2020.

[7] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015, pp. 3483–3491.

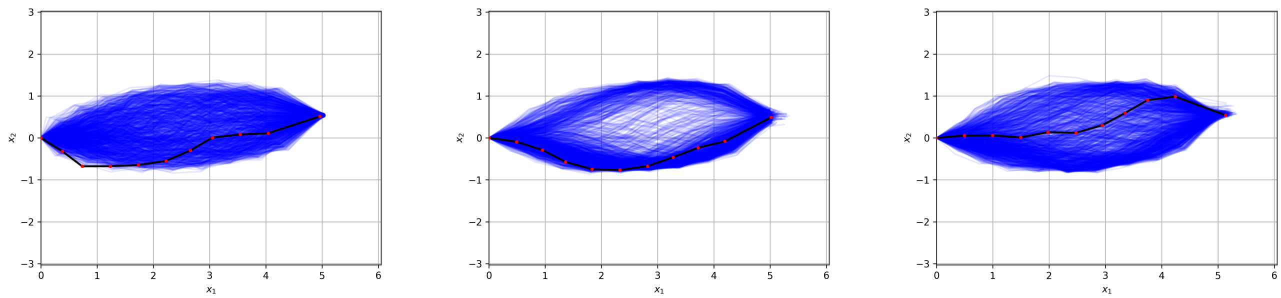
[8] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[9] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J.



(a) Rejection Sampling (b) cVAE (c) INN

Fig. 6. Arm configuration of a planar manipulator with 6 revolute joints and end-effector position at $(x, y) = [3.42, 0.68]$. 1000 samples are drawn from each model's predicted posterior $\tilde{p}(x|y_{gt})$, one random sample configuration is highlighted. $e_{posterior} = 0.012$ for the cVAE and $e_{posterior} = 0.0068$ for the INN.



(a) Rejection Sampling (b) cVAE (c) INN

Fig. 7. Arm configuration of a planar manipulator with 10 revolute joints and end-effector position at $(x, y) = [5.00, 0.53]$. 1000 samples are drawn from each model's predicted posterior $\tilde{p}(x|y_{gt})$, one random sample configuration is highlighted. $e_{posterior} = 0.057$ for the cVAE and $e_{posterior} = 0.025$ for the INN.

Smola, "A kernel method for the two-sample problem," *CoRR*, vol. abs/0805.2368, 2008. [Online]. Available: http://arxiv.org/abs/0805.2368

[10] Y. Zhou, C. Barnes, L. Jingwan, Y. Jimei, and L. Hao, "On the continuity of rotation representations in neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] J. Kruse, L. Ardizzone, C. Rother, and U. Köthe, "Benchmarking invertible architectures on inverse problems," Tech. Rep. i, 2019.

[12] B. Choi and C. Lawrence, "Inverse kinematics problem in robotics using neural networks," Tech. Rep., 1992.

[13] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," *CoRR*, vol. abs/1605.08803, 2016. [Online]. Available: http://arxiv.org/abs/1605.08803

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.