Milestone Report

Franziska Schwaiger
Matriculation number: 03658670

Thomas Brunner
Matriculation number: 03675118

Introduction

As explained in the proposal, 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 since the start and discuss the results we have obtained. Lastly, we explore possible next steps for our project.

Methods

Robot Simulations

For this project, we have implemented simulations of robotic arms with different complexities. In total, we have four simulations of planar robots with open kinematic chains. Three of these are composed solely of revolute joints and are of increasing complexity (with two, three and four joints). The fourth simulation also contains a prismatic joint. However, we decided to focus only on robot simulations with revolute joints and, thus, we limit ourselves to the first three simulations in this report. From now on, these simulations will be designated by their degrees of freedom (2 DOF, 3 DOF and 4 DOF). For an illustration of the robot configurations, refer to Figure 1.

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, both the 3 DOF and 4 DOF robot arms 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 one million samples of robot configurations for each robot simulation (2 DOF, 3 DOF and 4 DOF). Figure 1 shows an illustration of the datasets. Each configuration was sampled from a normal distribution $\theta_i \sim \mathcal{N}(\mu = 0, \sigma = 0.5)$. 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.

Conditional Variational Autoencoder

In the lecture, we have already discussed Variational Autoencoders (VAE) [3] in detail. One drawback of this type of generative architecture is that there is no control over the data generation process. In our case this would mean that although we can generate random sample configurations x we cannot control in which end-effector position y these configurations would result. Conditional Variational Autoencoders (cVAE) [4] solve this problem by conditioning the latent space z. For inverse kinematics this means that random samples are drawn from $p(z) \sim N(0, 1)$ and the predicted posterior of the joint angles is then generated conditioned on the end-effector position. During training, y is concatenated with both x and z and then fed into the encoder and decoder, respectively.

The cVAE is trained in the same manner as the VAE based on the ELBO (Evidence Lower Bound) loss $L_{ELBO} = L_y + L_x$:

$$L_y = -KL[q_{w_{enc}}(z|x, y)||p(z) \sim N(0, 1)] \tag{1}$$

The Kullback-Leibler divergence measures the distance between the predicted probability distribution $q_{w_{enc}}(z|x, y)$ of the latent space z and the standard normal distribution. The reconstruction loss is defined as follows:

$$L_x = E_{q_{w_{enc}}(z|x,y)}[\log(p_{w_{dec}}(x|z, y))] = \sum_{i=0}^{N} MSE(v_i \cdot \tilde{v}_i, 1) \tag{2}$$

Here, N is denoted as the number of joints. For representing the joint angles of the robot, we use a vector-based representation: $V = (\sin(\theta), \cos(\theta))$ to avoid singularities at the boundaries of the joint angles θ. $v_i \cdot \tilde{v}_i$ is denoted as the scalar product between the normalized predicted posterior point estimate $\tilde{v}_i = (\sin(\tilde{\theta}), \cos(\tilde{\theta}))$ and the ground truth vector $v_i = (\sin(\theta), \cos(\theta))$ of the ith joint.

Invertible Neural Network

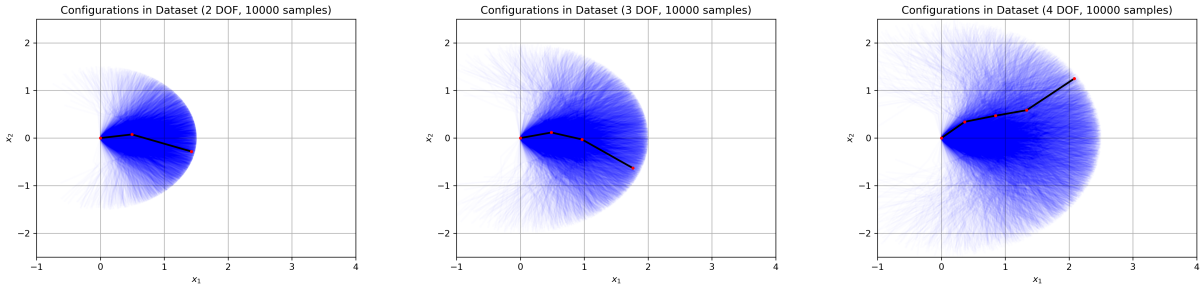
The motivation of Invertible Neural Networks (INN) [5] is to model a bijective mapping from the end-effector position to the joint angles. This is done by stacking alternately invertible blocks, so-called *coupling layers* [11] together with permutation layers (mix the data in a random but fixes way to enhance the interaction among individual variables). As every single unit is invertible, the whole INN is invertible, as well where the input x has the same dimensionality as the concatenated ouput $[y, z]$ to ensure bijectivity. The latent space z contains the information which is lost during the forward kinematics and not contained in y. The predicted posterior of the joint angles can be then generated in the same way as with cVAE by sampling from $p(z) \sim N(0, 1)$ and then predicting the joint angles conditioned on y.

The INN is trained based on four losses:

$$L_y = MSE(y_i, f_y(x_i)) \tag{3}$$

$$L_x = MSE(x_i, [f_y^{-1}(y_i), f_z^{-1}(f_z(x_i))]) \tag{4}$$

$$L_{p(z)} = MMD(p(f_z(x_i)), p(z) \sim N(0, 1)) \tag{5}$$



(a) 2 DOF (b) 3 DOF (c) 4 DOF

Fig. 1. Illustration of datasets used during training of models. Only a subset of the samples contained in the datasets is shown here. One configuration in the dataset is highlighted to illustrate the configuration of the robot arm.

$$L_{p(x)} = MMD(p([f_y^{-1}(y_i), f_z^{-1}(p(z))]), p(x)) \quad (6)$$

where we denote $[y, z] = [f_y(x), f_z(x)]$ as a bijective mapping between the input x (same vector-based representation of joint angles θ as for cVAE) and end-effector position y and latent space z. The losses (3) and (4) are computed by the Mean Squared Error (MSE) and losses (5) and (6) are computed by the Maximum Mean Discrepancy (MMD) [7]. This kernel-based method measures the distance between two distributions based on a batch of samples.

Implementation

The previous two model architectures (cVAE and INN) have been implemented in PyTorch and are inspired by existing implementations in [1], [2]. We utilized the Google Compute Engine to train our models.

Experimental Evaluation

Evaluation protocol

The two models have been trained and compared in a fair setting, with same batch sizes and approximately the same amount of trainable parameters. We used a train and test data splot of 70 and 30, respectively. For comparison, we used two metrics following the evaluation protocol of [9]:

1. The average mismatch between the true posterior obtained via rejection sampling $p_{gt}(x|y_{gt})$ and the predicted posterior $\tilde{p}(x|y_{gt})$ is computed for 100 randomly but fixed chosen samples from the test dataset. For each end-effector position of the subset of the test dataset, 100 samples are drawn from the true posterior by rejection sampling and compared with 100 samples predicted by the model via MMD.

$$e_{posterior} = MMD(\tilde{p}(x|y_{gt}), p_{gt}(x|y_{gt})) \quad (7)$$

2. The average re-simulation error is computed on the whole test dataset by applying the simulation for the true forward kinematics $y_{resim} = f(\tilde{\theta})$ to the generated joint angles $\tilde{\theta}$ and measure the mean squared distance between the ground truth end-effector position y_{gt} and the re-simulated end-effector position y_{resim} resulting from the predicted joint angles.

$$e_{resim} = E_{x\sim\tilde{p}(x|y_{gt})}(||f(x) - y_{gt}||_2^2) \quad (8)$$

Results

The results of the different robots are depicted in Tab. I. The results for the $e_{posterior}$ and e_{resim} are already quite good and in the same magnitude as in [5] and [9]. But in contrast to their results, the INN does not outperform the cVAE. One reason might be that none of the networks is fully optimized with respect to their hyper-parameters. Additionally, the training of the INN is more unstable than the training of the cVAE, the learning rate needs to be decreased by a factor of 10. In Fig. 2 and Fig. 3, the predicted and ground truth distributions of the joint angles given a specific end-effector position are shown for a 3 DOF and 4 DOF robot, respectively.

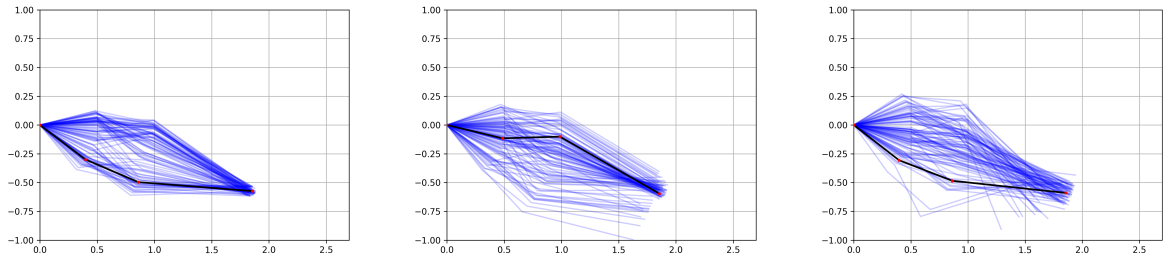
DOF	$e_{posterior}$	e_{resim}	Trainable Parameters	Model
2	0.077	**0.003**	164,808	cVAE
3	**0.045**	0.045	370,214	
4	0.063	**0.006**	373,220	
2	**0.061**	0.012	169,632	INN
3	0.066	**0.036**	369,660	
4	**0.044**	0.075	374,960	

TABLE I
RESULTS FOR THE CVAE AND INN WHICH HAVE BEEN TRAINED ON A PLANAR ROBOT WITH N DOF WITH $N = [2, 3, 4]$.

Conclusion

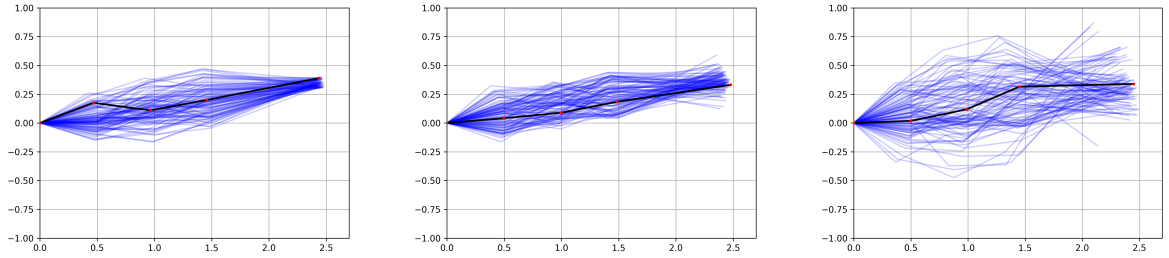
In this work, we have introduced the methods and robot simulations used to evaluate the feasibility of using neural networks for inverse kinematics problems in robotics. We have shown that the networks are able to learn the inverse transformations of planar robot simulations.

Our next steps will be to apply the networks to more complex and challenging scenarios, with the goal of evaluating the limits of the INN and cVAE architectures for inverse kinematics problems. In a first step, we will apply these networks to planar robots with longer kinematic chains and more degrees of freedom. As was previously discussed, our current setup uses robot simulations composed of revolute joints. In the future we plan on also using prismatic joints to evaluate the performance of the networks for different joint configurations. We also plan on eventually moving to three-dimensional space to evaluate the performance of these



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

Fig. 2. Arm configuration of a planar manipulator with 3 revolute joints and end-effector position at $(x, y) = [1.83, -0.57]$. 100 samples are drawn from each model's predicted posterior $\tilde{p}(x|y_{gt})$, one random sample configuration is highlighted.



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

Fig. 3. Arm configuration of a planar manipulator with 4 revolute joints and end-effector position at $(x, y) = [2.44, 0.35]$. 100 samples are drawn from each model's predicted posterior $\tilde{p}(x|y_{gt})$, one random sample configuration is highlighted.

networks on a higher dimensional space. As stated previously, the performance of our INN model is somewhat worse than the performance described in the paper [5]. One possible explanation for this mismatch is a suboptimal choice of hyperparameters. Thus, to improve the performances of our models, we plan on implementing hyperparameter optimization using random search.

REFERENCES

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

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

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

[4] 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.

[5] 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

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

[7] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample problem," *CoRR*, vol. abs/0805.2368, 2008. [Online]. Available: http://arxiv.org/abs/0805.2368

[8] 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.

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

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

[11] 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