Standard Deep Generative Models for Density Estimation in Configuration Spaces: A Study of Benefits, Limits and Challenges

Robert Gieselmann and Florian T. Pokorny

***Abstract*— Deep Generative Models such as Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE) have found multiple applications in Robotics, with recent works suggesting the potential use of these methods as a generic solution for the estimation of sampling distributions for motion planning in parameterized sets of environments. In this work we provide a first empirical study of challenges, benefits and drawbacks of utilizing vanilla GANs and VAEs for the approximation of probability distributions arising from sampling-based motion planner path solutions. We present an evaluation on a sequence of simulated 2D configuration spaces of increasing complexity and a 4D planar robot arm scenario and find that vanilla GANs and VAEs both outperform classical statistical estimation by an n-dimensional histogram in our chosen scenarios. We furthermore highlight differences in convergence and noisiness between the trained models and propose and study a benchmark sequence of planar C-space environments parameterized by opened or closed doors. In this setting, we find that the chosen geometrical embedding of the parameters of the family of considered C-spaces is a key performance contributor that relies heavily on human intuition about C-space structure at present. We discuss some of the challenges of parameter selection and convergence for applying this approach with an out-of-the box GAN and VAE model.**

I. Introduction

The successful estimation of probability distributions in robot configuration space $\mathcal{C}$ is of fundamental importance for many applications within the framework of probabilistic robotics [1] and sampling-based robot motion planning. Sampling-based motion planners (SBMPs) [2], in particular benefit from heuristic sampling distributions that emphasize sampling within certain regions of the configuration spaces in order to speed up planning. However, hand-crafting such heuristics is challenging as the C-space complexity is increased. Furthermore, the goal is typically to understand not just a single distribution on configuration space $\mathcal{C}$ for a fixed start and goal configuration, but a family $\mathcal{F}$ of C-spaces $\mathcal{C}_\lambda$ and associated probability density functions p_λ for each λ in some indexing set S.

$$\mathcal{F} = \{(\mathcal{C}_\lambda, p_\lambda) : \lambda \in S, p_\lambda : \mathcal{C}_\lambda \rightarrow \mathbb{R}_{\geq 0}, \int_{\mathcal{C}_\lambda} p_\lambda(x)\mathrm{d}x = 1\} \quad (1)$$

For example, one may want to estimate the distribution of RRT solution trajectory nodes parameterized by start and end configurations for a particular fixed configuration space,

*This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation

Robert Gieselmann and Florian T. Pokorny are with RPL, EECS, KTH Royal Institute of Technology, Stockholm, Sweden `robgie, fpokorny@kth.se`

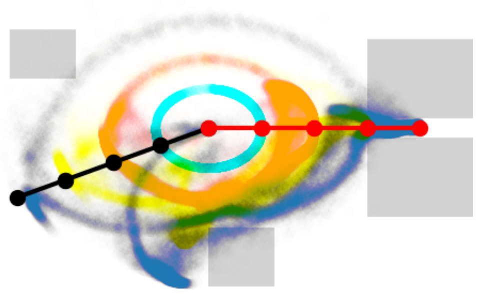
Fig. 1: A 4 DoF planar robot arm with sampling distributions parameterized by arbitrary end-effector start (x, y) configuration and fixed goal state in between the rectangles to the right resulting in a family $\mathcal{F}$ of disctributions in 4 dimensional C-space. We illustrate one such distribution for a given starting configuration the bottom left. Colors indicate regions of high density for samples estimated by a Generative Adversarial Network and projected into the workspace.

or one may be interested in understanding sampling distributions for continuously parameterized obstacle positions or discrete door openings states. Fig. 1 shows such a planning scenario for a 4 DoF planar manipulator and densities for each joint angle estimated by a GAN.

In this work, we provide a first empirical study of the problem of density estimation with a vanilla GAN and VAE network structure on a selection of parameterized configuration spaces $\mathcal{F} = \{(\mathcal{C}_\lambda, p_\lambda) : \lambda \in S\}$, where $\mathcal{F}$ has dimension 2, 6, 8, 10 or 12, S is discrete or continuous, and each indexed $\mathcal{C}_\lambda$ either has dimension 2 (planar configuration space) or 4 (4 DoF robot arm case) respectively. The key findings of our study are:

i) GAN and VAE performance heavily depended on hyperparameters, with VAE being particularly sensitive to KL-rate settings on our problem domains.
ii) For our settings, GANs produced sharper density estimates, which resulted in fewer C-space collisions when the learned distribution was used for RRT sampling.
ii) The chosen featurization of S is a key factor determining the performance of GANs and VAEs. We propose a permutation-based baseline test to understand the impact of the choice of such embeddings.
iv) We conclude that, while out-of-the-box Deep Generative Models are a promising new tool for density estimation in C-space, key advances are needed to improve the stability and convergence of these methods for robust applications in motion planning.

II. Background and Related Work

Deep Generative models learn a representation of the underlying data distribution from a training set T and can be used to generate synthetic samples after training. In this work, we focus on applications of the two most prominent Deep Generative Models: Variational Autoencoders (VAE) [3] and Generative Adversarial Networks (GAN) [4], that arose in applications in computer vision. We employ the conditional extensions, the Conditional Variational Autoencoders [5] and Conditional Adversarial Generative Networks [6], in order to take possible parameterizations of the environment into account.

A Generative Adversarial Network [4] consists of two networks, the generator and the discriminator. Given a vector of random noise - usually sampled form a Gaussian or uniform distribution - the generator's task it to produce synthetic data samples. The discriminator decides whether or not those samples originate from the training distribution. Both generator and discriminator are represented by a neural network and during training, the generator attempts to fool the discriminator by generating fake samples which are indistinguishable from real samples. At the same time, the discriminator is trained to distinguish generated and real data. From a game-theoretic point of view the objective is defined as a minimax game while the optimum corresponds to the state where real and fake data becomes indistinguishable. The objective function is given by Eq. (2), where $D(z)$ corresponds to the discriminator and $G(x)$ to the generator network.

$$\begin{aligned}\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}}(x)[\log D(x)] + \\ \mathbb{E}_{z\sim p_z(z)}[1 - D(G(z))].\end{aligned} \tag{2}$$

A Variational Autoencoder [3], on the other hand, is composed of two neural networks which are called encoder and decoder. The encoder network maps a data sample to a lower-dimensional latent vector while the decoder attempts to reconstruct the original data sample given a latent representation. Starting from variational inference to approximate the distribution of latent variables given the data, the VAE models latent posterior and prior with Gaussians. Its objective function, shown in Eq. (3), has a sound probabilistic interpretation as it directly maximizes the evidence lower bound (ELBO).

$$\begin{aligned}\mathcal{L}(\theta,\phi;x) = \mathbb{KL}[q_\phi(z|x)||p_\theta(z)] - \\ \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)].\end{aligned} \tag{3}$$

The posterior $q_\phi(z|x)$ corresponds to the encoder network, whereas the likelihood $p_\theta(x|z)$ is represented by the decoder network. In the standard setting, the logarithmic likelihood $\log p_\theta(x|z)$ is maximized by minimizing the mean-squared error (L2) between the reconstruction and the ground truth observation. One can think of the Kullback-Leibler divergence in Eq. (3) as a regularizing term which enforces the distribution of each latent dimension to be Gaussian. This has the advantage that the generation of data points is reduced to sampling from a Gaussian and feeding the obtained latent code z through the decoder network.

In this work the use a further modification of the Variational Autoencoder called β-VAE [7]. It introduces a weighting factor β which balances the regularization of the latent space. It is often used to disentangle latent variables which helps to understand the semantics of different latent dimensions. Notice that it introduces a trade-off between allowing complex latent representations and quality of generated data points. Throughout the following, we refer to this β-VAE simply as "VAE", as a shorthand.

Applications of these generative methods in Robotics are as of now still limited compared to the now classical Deep Convolutional Neural Networks that are widely evaluated and applied [8], [9]. Early applications of GANs in Computer Vision in particular have gained much attention by demonstration of impressive results in generating synthetic photorealistic images [4]. In this work, we consider the use of these techniques to model distributions in robot configuration spaces. At this point it is not fully clear whether the promising results in high-dimensional settings of the space of images can transfer to equally important breakthroughs in robotics applications involving data in robot configuration spaces, where hard obstacle and joint constraints need to be respected.

A few recent publications have started to focus on the problem of estimating various sampling distributions in configuration space using deep generative models in order to improve the efficiency of sampling-based motion planners. The authors of [10] attempt to learn the distribution of narrow passages using a Conditional VAE. Narrow passages usually represent a challenge as the chance of sampling within those regions is relatively low due to their small volume. Again, the neural network is applied to decrease the amount of exploration for a sampling-based planner. [11], uses a Conditional VAE to learn a sampling distribution for robot path planning. Given the current configuration, obstacles and goal, the model predicts the distribution of states along the shortest path. [12] present an algorithm which integrates a feed-forward neural network into a sampling-based motion planner. Given the current configuration, the goal and the environment parameters, the model predicts the location of the next sample along the optimal path towards the goal. By using dropout during the test phase, entire trajectory rollouts can be generated which are used to initialize the states of the search tree. The authors show that the methods significantly reduced the planning computation time for a six-dimensional articulated robot. In contrast to the previous approaches, [13] employ a generative adversarial network to learn an action sampler to map a state to a distribution of states given the problem description. The authors motivate the use of GAN compared to other methods by the fact that it does not require to explicitly define a distance measure between synthetically generated and ground truth samples.

In the above mentioned works, the focus is on the establishment of novel applications of generative models to the domain of motion planning and results are not benchmarked against statistical density estimation methods. We instead study the convergence of these methods with respect to a simple baseline histogram density estimator and consider both standard (rather than domain specific and specialized) Conditional GAN and VAE architecture for evaluation.

One of the open questions with respect to GANs and VAEs which we study in this paper is to what extent these methods can take advantage of geometry encoding and problem specific properties to beat an off-the-shelve histogram based approach for density estimation in robot configuration spaces.

III. Methodology

For each of the following experiments, we consider a specific family $\mathcal{F}$ of configuration spaces $\mathcal{C}_\lambda$ and probability density $p_\lambda : \mathcal{C}_\lambda \to \mathbb{R}$, for λ in some indexing set S. In the experiments, $p_\lambda(x)$ arises as the probability that $x \in \mathcal{C}_\lambda$ occurs as a node in a found RRT [14] (or RRT-Connect [15]) solution trajectory for a specific motion planning problem on $\mathcal{C}_\lambda$ by the following process: We first generated a training dataset $\mathcal{T}$ by uniformly sampling $\lambda \in S$ k_S times then adding all nodes of k_R independent successful RRT solution runs for that parameter to the training dataset.

As ground truth estimate for the underlying density generating these points in configuration space, we utilize histograms of fixed bin resolution. We then trained both Conditional GANs and Conditional VAEs with conditioning on $\lambda \in S$. When a particular λ parameter is given, this then enables us to create new samples from the learned estimate $\hat{p}_\lambda$ for p_λ. We implemented all models in PyTorch [16] and trained them on a single GPU. For the experiments in section IV-A we used 5 fully connected layers with 256 neurons per layer for generator and discriminator network each for the considered GANs, while the VAEs used also 5 fully connected layers with 256 neurons per layer for each the encoder and decoder. Later, for the experiments in sections IV-B, IV-C and IV-D, we increased the number of layers to 7 for each the generator and discriminator, respectively encoder and decoder networks. For all experiments we used latent dimensions of 16. We utilized leaky ReLU activation functions for each and trained the networks using Adam [17] for GANs and Adagrad [18] algorithm for VAEs. We evaluated multiple different and repeated hyperparameter settings for each model based on prior experience and focus the presentation of results on the best-performing models.

To measure the accuracy of the approximated distributions we compute the Wasserstein metric, which provides a natural earth mover-type notion of similarity between densities. We utilized the implementation found in [19] to compute distances between density histograms. Other evaluation criteria used in this work are the mean squared-error distance, Bhattacharyya distance and relative number of collisions with obstacles. Lastly, we suggest comparing the trained models in the context of an application. This was done by adding a sampling distribution utilizing the trained GAN/VAE directly within the RRT-Connect implementation of the Open Motion Planning Library (OMPL) [20] and evaluating the characteristics of the resulting paths.

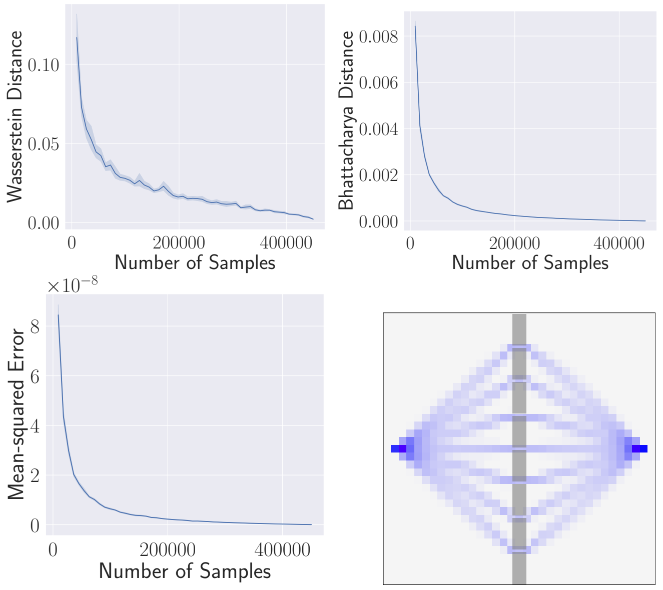

Fig. 2: Convergence of the Wasserstein, Bhattacharyya and mean-squared error distance with respect to the full data set over increasing number of samples. Bottom right: histogram of ground truth density. Dark gray boxes indicate obstacles.

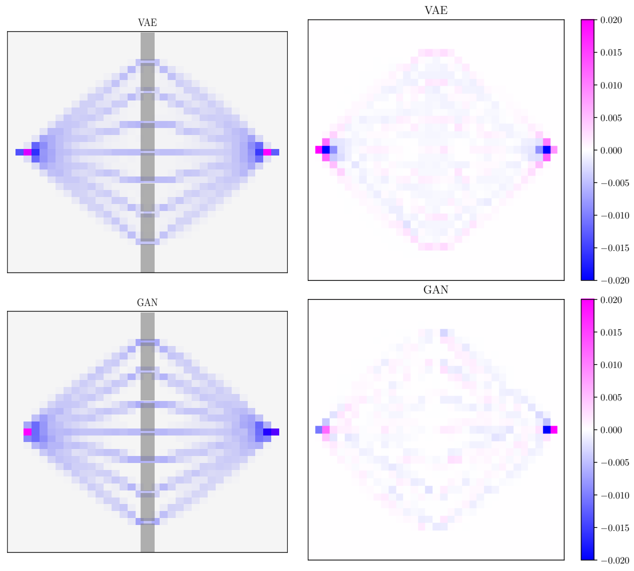

Fig. 3: Left column: estimated densities using VAE and GAN. Right column: error between ground truth and estimated density histograms

IV. Experiments

A. Static Planar Environment and Histogram Parameters

Here we study a simple planar C-space independent of a λ parameter and consisting of a box with corners $(-10, -10)$ and $(10, 10)$ and a wall along $x = 0$ with the indicated 7 narrow passages as shown in the right bottom of Fig. 2. A probability density p of samples was generated by randomly opening one of the narrow passages, determining $k_R = 20$ independent RRT* solution and adding the found RRT* nodes to the sample set. In this manner we generated a training set with 469279 samples from 5000 RRT* runs. As baseline, we work with perhaps the most classical density estimation technique [21] available: a simple normalized

TABLE I: Results of Experiment in Static Environment

Criterion	Variational Autoencoder	Generative Adversarial Net
Wasserstein Distance	$8.21\text{e}{-2} \pm 2.65\text{e}{-3}$	$8.57\text{e}{-2} \pm 6.53\text{e}{-3}$
Mean-Squared Error	$2.92\text{e}{-7} \pm 6.23\text{e}{-9}$	$2.53\text{e}{-7} \pm 6.38\text{e}{-9}$
Bhattacharyya Distance	$1.46\text{e}{-2} \pm 1.43\text{e}{-4}$	$1.06\text{e}{-2} \pm 1.83\text{e}{-4}$
Collision Ratio	$1.40\text{e}{-2} \pm 2.04\text{e}{-4}$	$2.67\text{e}{-3} \pm 1.20\text{e}{-4}$

n-dimensional histogram with fixed resolution, which, for each n dimensional bin B, approximates the probability of a sample falling into B by the ratio of samples in B divided by the total number of samples. Recall that, for i.i.d. sampled data the central limit theorem guarantees convergence of the mean probability over each bin to the histogram estimate with convergence rate $O(\frac{1}{\sqrt{n}})$ with n samples. While this is in some sense the best one can hope for for arbitrary sample densities, explicit discretization into bins of small volume is in practice infeasible for dimensions higher than 3, or when working with a large family of 2D configuration spaces.

As ground truth estimate, we compute a histogram approximation to p from the training data with an even splitting of the configuration space domain into 35×35 cells. Fig.2 displays how quickly the histogram estimate for this chosen discretization into bins approaches to the histogram estimate for the full training dataset with 400.000 samples, providing us with a sense of the relationship between number of samples and expected error. For example, with 100.000 samples we can expect an accuracy of approximately 0.028 with respect to the Wasserstein distance between the density histograms. Similarly, a convergence test with respect to Mean Squared Error and Bhattacharyya distance indicated a precision of approximately $6.4\text{e}{-9}$ and $6.4\text{e}{-4}$ for 100.000 samples with respect to these metrics.

Next, we trained both a GAN and VAE with a total of 10 fully connected layers of 256 neurons each as described in the Methodology section to confirm convergence in this very simple setting. Both models approximated the ground truth distribution well as shown in Table I. In terms of remaining Wasserstein error, we could not observe significant performance differences between GAN and VAE. The two plots in Fig. 3 show the difference between ground truth histogram and estimated distributions. We found the predicted distribution of the VAE to be slightly more blurry generating samples in regions between the passages. This also provides an explanation for the higher number of collision with workspace obstacles using this method.

In image synthesis, VAEs are known to produce blurry images [22]. According to [23] this originates from the VAE approximation of the maximum likelihood. As a consequence of using the L2 reconstruction in the objective, the distribution of samples that map to the same latent code is approximated with a fixed variance Gaussian. Due to overlapping latent representations, i.e. the encoder mapping different x to the same latent code z, the optimal decoder produces an averaging of points in the observations space which manifests itself in fuzzy reconstructions. This overlap inevitably happens as a consequence of Gaussian encoders

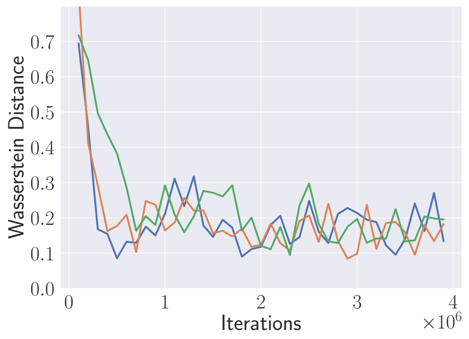

Fig. 4: Wasserstein error of the GAN for serveral training runs using the same hyperparameters. Note that the model converges rapidly until 500k iterations and stabilizes thereafter.

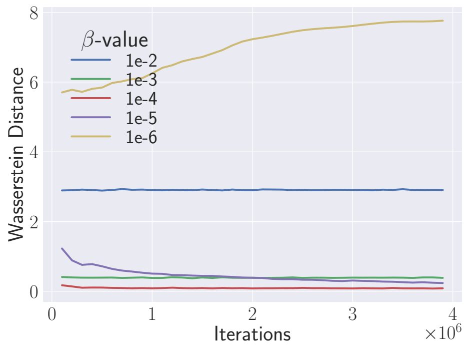

Fig. 5: Wasserstein error for VAE trained with different values for parameter β. The model achieved the best performance for a low value of $\beta = 1\text{e}{-4}$.

$q_\phi(z|x)$ [24].

The training of the GAN was generally more unstable and highly sensitive regarding the chosen learning rates and batch size. We found that small learning rates of $1\text{e}{-5}$ for the generator and discriminator networks and large batch sizes greater than 256 often increased the stability. Compared to the VAE, the performance after being trained with the same parameters varied enormously. The plot in Fig. 4 illustrates this by showing the Wasserstein error during training for several runs. Unlike the GAN, we found the performance changes between repeated VAE training runs to be negligible. Further, the VAE produced stable learning updates and usually converged quickly. We observed that choosing the proper β-value, i.e. regularization of Kullback-Leibler loss, is crucial for achieving decent results using the VAE (see Fig. 5). A value of $1\text{e}{-4}$ provided the best accuracy for this experiment. Generally, smaller values resulted in better training while choosing it too small, in our case lower than $1\text{e}{-5}$, significantly slowed down the training.

B. Parameterized Door Environments - Natural Embedding

In this set of experiments, we consider a family of configuration spaces $\mathcal{C}^k_\lambda$, in an environment with $k = 4, 6, 8, 10$ vertical obstacle walls that each contain 8 doors. For each λ exactly one door per layer is opened and the resulting density p_λ is generated by nodes of solution trajectories through exactly these doors. Note that there are 8^k possible door opening parameters, creating an environment with a combinatorial complexity challenge. The initial and goal

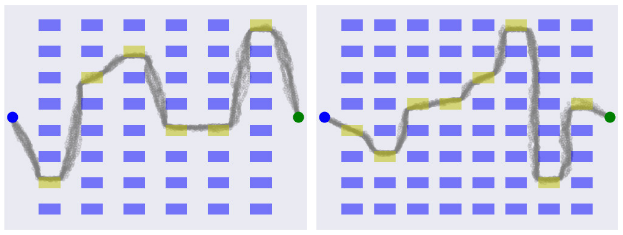

Fig. 6: Examples of two environments with $k = 6$ and $k = 8$ respectively. Ground truth densities for the λ settings of particular door openings are shown in dark gray. Blue rectangles indicate closed passages, yellow rectangles show open passages.

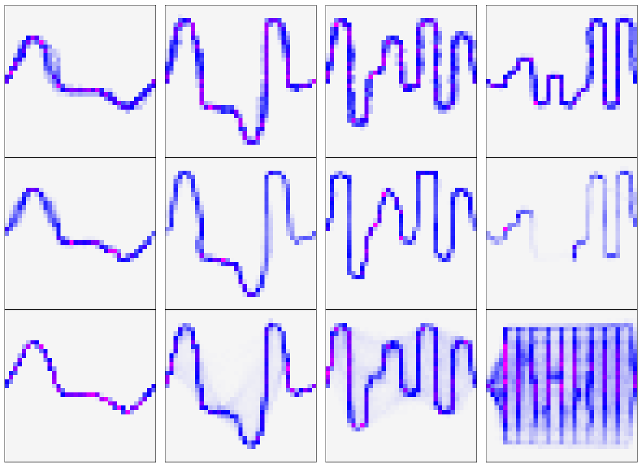

Fig. 7: Examples for ground truth (top row) and estimated distributions using GAN (second row) and VAE (third row) for test data for $k = 4, 6, 8, 10$ (left to right). Both models generated accurate estimates up to $k = 8$ while VAE estimates were thoroughly blurry.

configurations are defined to be at $(-9.9, 0)$, respectively at $(9.9, 0)$. The training samples were generated by evaluating RRT* 10 times for a particular choice of λ. For each environment k we generated a training data set using 10000 randomly sampled λ. This yields to 100000 sampled trajectories in total per environment. Figure 6 shows an illustrative example of the workspace for $k = 6$, respectively $k = 8$, and ground truth densities.

As is often the case in Robotics, training data is computationally expensive to obtain even in this simulated setting and thus covers only a fraction of the space of possible observations. While the training set for $k = 4$ contains all 8^4 combinations of possible λ, we have to deal with an exponentially increasing number of unseen configurations in the case of $k = 6, 8, 10$. For instance, if $k = 8$ the model experiences merely 0.06% of possible door openings during training. One part of this section investigates to what extent the learned models are able to interpolate unseen samples. We employed a natural embedding where each vertical layer is represented by a one-hot vector indicating which of the passages in that layer is open. For each k we train a VAE and GAN model using hyperparameters based on preliminary evaluations and insights from the section 1. The test sets are each composed of 1000 randomly selected λ that were not observed during training (except for $k = 4$).

Fig. 7 presents examples of estimated conditional distributions for $k = 4, 6, 8, 10$. In addition, Fig. 8 shows results

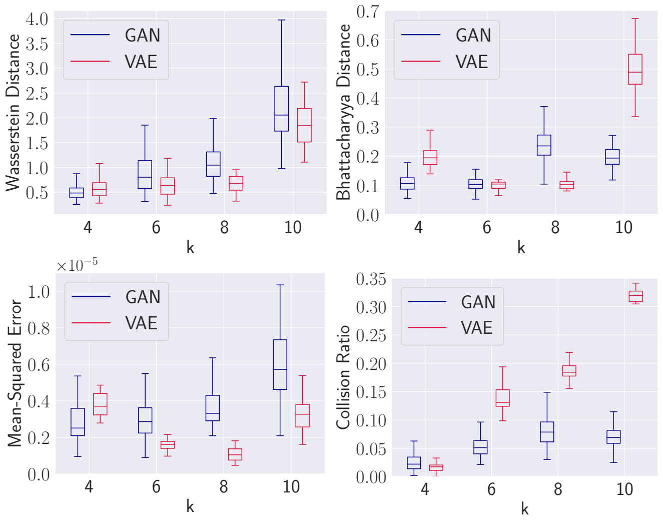

Fig. 8: Average Wasserstein, Bhattacharyya, mean-squared error distance and collision ratios on test sets for GAN and VAE with respect to different parameters k. Overall, VAE achieved slightly lower error values than GAN while the blurriness of its estimates yields to a significant increase of collisions in more complex environments.

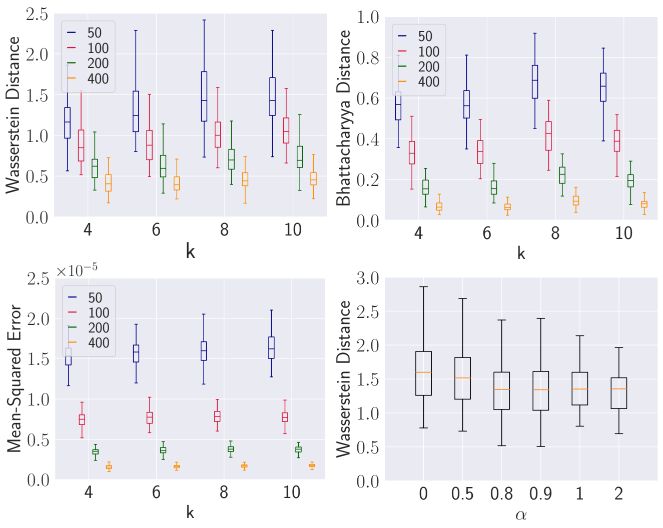

Fig. 9: Average Wasserstein, Bhattacharyya, mean-squared error distance for environments k using subsets of different size sampled from the ground truth data. Bottom right: Wasserstein distance of using a weighted K-Nearest Neighbor estimator with respect to hyperparameter α. The KNN estimator achieved the lowest average remaining Wasserstein distance of 1.35 for the choice of $\alpha = 0.8$.

of the numerical evaluation and displays the remaining distances with respect to the ground truth density histogram. Our generative models produced visually plausible density estimates up to $k = 8$. Except for $k = 4$, VAE outperformed GAN in terms of Wasserstein, Bhattacharyya and mean-squared error distance. Again, we observe the tendency VAE to produce fuzzy samples. This issue appears to get stronger under more complex environment conditions and results in an increasing number of collisions. The GAN estimates are sharper, however it highly overestimates density in some regions along the paths (see Fig. 7). We attribute this finding to mode collapse, a commonly observed instability in GAN training where generated samples lack diversity. For $k = 10$ we observed a strong performance drop for both models in terms of Wasserstein distance with respect to the ground truth histograms. For the GAN we observed that entire regions of the distribution were missing. Possible explanations are insufficient amount of training examples and the aforementioned training instability.

In addition to training our generative models, we further evaluated the performance of a simple histogram estimator of size 32×32. It can be thought of as a lookup generated from a limited amount of ground truth data points. We created such histograms for each configuration λ using either 50, 100, 200 or 400 samples. The corresponding numeric results are shown in Fig. 9. Histograms with 400 samples clearly outperformed the trained generative models. However, one has to consider what 400 samples mean in terms of required memory. Given $k = 8$ and that sample points are represented as float values, the amount of memory needed to store 400 samples for all 8^8 distinct λ exceeds 53 gigabytes. Note that storing the weight and bias parameters of the neural networks takes up less than 1.4 megabyte for the generator (GAN), respectively decoder network (VAE). During GAN/VAE training the equivalent number of samples that were observed per possible door combination for $k = 6$ is roughly 100. The corresponding histogram estimator demonstrated worse performance than the GAN and VAE for this number of samples (see Fig. 8 and 9).

The previous histogram estimator has direct access to all ground truth densities. In order to establish a fair comparison with classical tools, we benchmarked a K-nearest neighbor (KNN) estimator which makes predictions solely based on a database of experienced densities. Given a query test configuration, the KNN method estimates the density by computing a weighted average of the K-nearest neighbor histograms in the training set. We defined the distance by means of the squared Euclidean distance between the vector embeddings of different λ and employed an exponential kernel to weight neighbors. This method interpolates the training samples and does not require any training apart from tuning a weighting parameter α. Given an unseen configuration v and the experienced K nearest neighbors w_i, the density histogram $H_{\text{estimated}(v)}$ is estimated by

$$H_{\text{estimate}}(v) = \frac{\sum_{i=1}^{K} e^{-\alpha \|v-w_i\|^2} H(w_i)}{\sum_{i=1}^{K} e^{-\alpha \|v-w_i\|^2}} \tag{4}$$

We carried out this comparison for an environment with 8 vertical layers and tested a broad range of values for parameter α. The bottom right plot in Fig. 9 presents the resulting average distances for the K-nearest neighbor estimator considering $K = 20$ nearest neighbors. A value of $\alpha = 0.8$ led to the lowest average remaining Wasserstein error of approximately 1.35, significantly higher than for the discussed deep generative models. Note that $\alpha = 0$ corresponds to the unweighted average of all 20 nearest neighbors while a greater α increased the impact of the 1-nearest neighbors. The histograms in Fig. 10 illustrate this dependency and also show a qualitative comparison to the estimates of GAN/VAE.

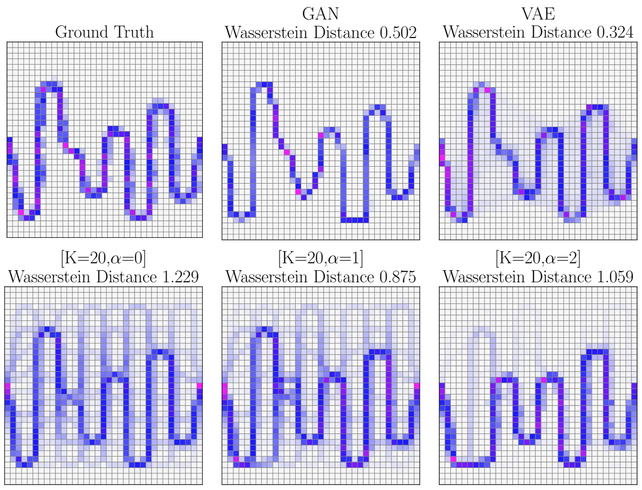

Fig. 10: Top row: Density histograms for ground truth and GAN/VAE estimates. Bottom row: weighted KNN estimates ($K = 20$) and different α. Overall, the trained deep generative models provided better estimates measured in terms of remaining Wasserstein distance than a KNN-based histogram estimator.

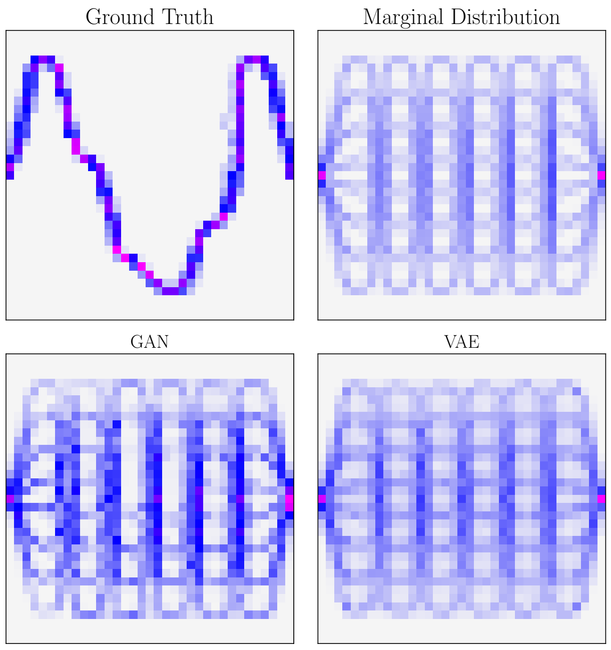

Fig. 11: Example of density histograms for ground truth, GAN and VAE for randomized embedding ($k = 8$). Also density of ground truth marginal distribution. The generative models did not provide plausible estimates for this setting. Seemingly, the learned estimates resemble some average over the set of all possible door parameterizations.

C. *Parameterized Door Environments - Randomized Embedding*

We now test the model performance influence of the geometric encoding of the door opening parameter λ by means of the previous 1-hot embedding that utilized the same ordering of 1-hot vector dimensions as the vertical order of the doors. For this purpose, we repeated the experiments from the previous section using a randomized embedding of λ to parameterize the configuration spaces. We generated the embedding by randomly shuffling the previously used parameterizations, i.e. we randomly assign a one-hot vector to each conditional distribution in such a way that no two different environments share the same encoding. We trained both GAN and VAE with the same hyperparameters as before for an environment with $k = 8$.

The three histograms in Fig. 11 qualitatively show the estimated densities after training. It can be seen that both GAN and VAE were not able to identify the correct environment configuration. It seems that the capability of recognizing configurations is therefore highly dependent on the choice of λ embedding. Accordingly, the average Wasserstein distances were 3.0 ± 0.75 for GAN, respectively 2.98 ± 0.74 for VAE as compared 1.05 ± 0.35 and 0.71 ± 0.35 when utilizing the non-randomized embedding. We do not find any differences between train and test performances while both models predicted some fuzzy mixture over all possible density conditionals. This is somewhat surprising, since the models did not attempt to memorize observed samples given these infeasible circumstances. Fig. 11 further depicts an approximation of the density histogram of the marginal distribution of data points, i.e. the ground truth distribution integrated over all parameterizations λ (assuming the same probability for all λ). The VAE estimates a similar distribution averaging over all possible λ which seems to be a consequence of its reconstruction objective. The GAN estimate is to some extent similar while, again, we identified an exaggerated strengthening of some modes of the distributions, while some other modes were missed by the GAN estimate, possibly due to the mode collapse phenomenon previously observed for GANs in Computer Vision settings.

D. Parameterized Robot Arm Environment

Robot configuration space distributions are often parameterized by a number of continuous variables, e.g. the starting coordinates of the robot or joint angles. In this section, we consider a 4 DOF planar robot arm with sampling distributions parameterized by an arbitrary start configuration of joint angles $(\theta_0, \theta_1, \theta_2, \theta_3)$. The goal configuration is set to be at $(0, 0, 0, 0)$. The manipulator starts at an initial configuration $(\theta_{init}, 0, 0, 0)$ where θ_{init}, the base joint angle, ranges from $-\pi$ to π. The environment contains 4 rectangular shaped obstacles as shown in Fig 12 while 2 of them form a narrow passage in the workspace around the goal.

We are now interested in estimating the distributions of samples generated from solutions of a sampling-based planner for the presented robot and workspace. We trained Conditional GANs and VAEs to estimate densities conditioned on the initial joint angle of the robot. Again, we applied the same network sizes and architectures as in the previous sections IV-B and IV-C. Instead of RRT* we used RRT-Connect for this experiment which concurrently grows two trees from the start to the goal and vice-versa, in order to speed up planning through the narrow passage. Given the current setup and the inherent stochasticity of the planner, the generated paths can deviate largely in terms of shape and length even for similar initial configurations, illustrating an important property of such distributions in the robotics context as illustrated in the top in Fig. 12. We created a training set by running the planner 3 times for each of 26000 randomly sampled θ_{init}. In a similar fashion, we generate a test set using 1000 distinct θ_{init}.

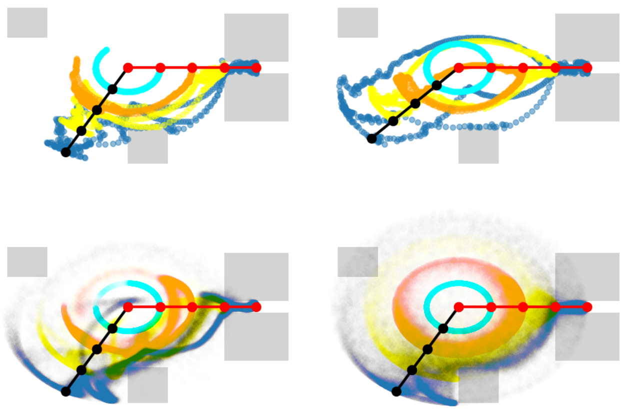
Fig. 12: Top row: Illustration of workspace and planer robot arm. The light gray boxes indicate obstacles. The initial and goal configurations are colored in black, respectively red. Remaining colors indicate distributions of joint nodes (θ_0:cyan, θ_1:orange, θ_2:yellow, θ_3:blue) given a particular start configuration. Bottom: Estimated densities for GAN (left) and VAE (right). Corresponding ground truth data is presented in the top left plot.

We found that the GAN produced significantly 'sharper' densities (transformed in workspace coordinates) than VAE which is exemplified by the bottom plots in Fig. 12. The assessment of estimates with respect to the ground truth densities is less suitable for this experiment due to the strong stochasticity and the low number of samples per configuration.

Instead, we directly measured the utility of the estimators and used them as sampling heuristics to guide the space exploration during planning with RRT-Connect for the presented environment. We analyzed the quality of the resulting paths by looking at criteria such as the number of states until a valid solution was found (same as number of collision checks), the number of collisions, the path length and the number of points of which the solutions consist of. The curves in Fig. 13 show the average values for those criteria with respect to the test set. The x-axis represents the influence of the bias β of the sampling heuristic. The motion planner samples from our learned distribution with bias probability β, with probability 0.05 the goal state and a uniform random samples otherwise. We compared both GAN and VAE with respect to two baselines. Firstly, a uniformed sampler which resembles the standard RRT-Connect implementation. Secondly, a lookup table containing the ground truth from 3 RRT-Connect runs. We observed a strong influence of the bias values which tunes the influence of the heuristic. As shown, GAN outperforms VAE with respect to the average required number of states. Most of the time, VAE required more time to find a valid solution which is related to the higher number of collisions. Since it used a larger number of states to assemble the path, it yielded to shorter paths and higher number of nodes per path. The GAN focused sampling in the most frequent

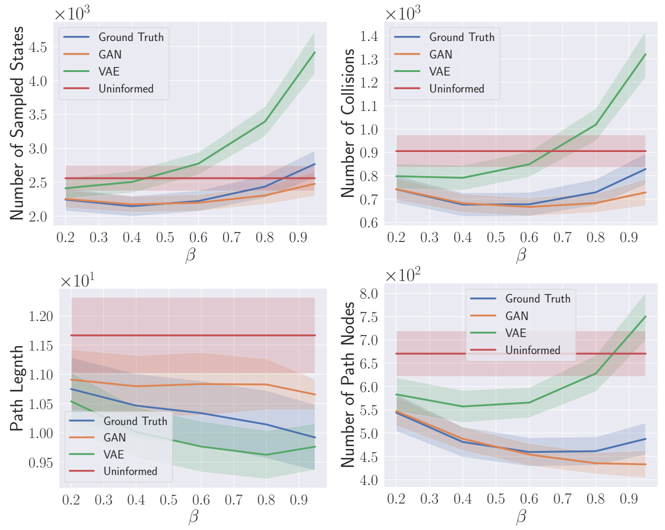

Fig. 13: Average values for number of states, number of collisions, path length and size of solution path over heuristic bias β evaluated on the test set.

regions while the VAE models spreads samples more evenly over the space. Due to the diversity of possible paths the resulting densities are highly multimodal and it appears that the aforementioned shortcoming of VAEs, originating from Gaussian encoders and it's objective, may pose an obstacle towards the application of VAEs in such environments.

Interestingly, we observed that a certain amount of random sampling helped to speed up the planning. This can be seen by looking at the respective curves of the GAN and ground truth sampler, which both show a performance decline for bias value greater 0.4.

V. Conclusions and Future Work

We presented a comparative study discussing the challenges, benefits and limits of using state-of-the art deep generative models, represented by a standard GAN and VAE architecture, for density estimation in robot configuration spaces. The main challenges we observed were tuning of hyperparameters, instabilities during training, missing modes for GANs and blurry VAE estimates. Our experiments showed that the typically manual and potentially overlooked choice of parameterized configuration space embedding can have a larger effect on performance than the choice of model. We believe the development of optimal such embeddings for particular robotics applications is an interesting direction for future work. Similarly, while classical techniques may not scale to high-dimensional configuration spaces, we recommend for future work on developing novel generative models in robotics to benchmark also against more classical density estimation techniques such as simple histograms or kernel density and nearest neighbor estimators in order to contribute to the community's understanding of relative performance characteristics of deep generative models for robotics.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

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

[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] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-vae:learning basic visual concepts with a constrained variational framework," *ICML*, 2017.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[9] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford *et al.*, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[10] R. Kumar, A. Mandalika, S. Choudhury, and S. Srinivasa, "Lego: Leveraging experience in roadmap generation for sampling-based planning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[11] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 2018, pp. 7087–7094.

[12] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[13] T. L.-P. Beomjoon Kim, Leslie Pack Kaelbling, "Guiding search in continuous state-action spaces by learning an action sampler from off-target search experience," in *Proceedings of the 32th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2018.

[14] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[15] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA.*, vol. 2, April 2000, pp. 995–1001 vol.2.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[18] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 07 2011.

[19] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, September 2009, pp. 460–467.

[20] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[21] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[22] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 658–666.

[23] S. Zhao, J. Song, and S. Ermon, "Towards deeper understanding of variational autoencoding models," *CoRR*, vol. abs/1702.08658, 2017. [Online]. Available: http://arxiv.org/abs/1702.08658

[24] O. Bousquet, S. Gelly, I. Tolstikhin, C. J. Simon-Gabriel, and B. Schölkopf, "From optimal transport to generative modeling: the vegan cookbook," Tech. Rep., 2017.