Predicting Collision Detection with a Two-Layer Neural Network

Tuan Tran[1], Jory Denny[2], and Chinwe Ekenna[1]

Abstract— **Fast and efficient motion planning algorithms are critical for many state-of-the-art robotics applications, such as self-driving cars or rescue operations. Existing motion planning methods become less effective as the dimensionality of the robot and complexity of the workspace increases, e.g., the computational cost of collision detection routines increases. In this work, we present a framework to address the cost of expensive primitive operations in sampling-based motion planning. This framework determines the validity of a sample robot configuration through a novel combination of a Contractive AutoEncoder (CAE), which captures a point cloud representation of the robot's workspace, and a feed-forward neural network, which efficiently predicts the collision state of the robot from the CAE and the robot's configuration. We evaluate our framework on multiple planning problems with a variety of robotics in 2D and 3D workspaces. The results show that (1) the framework is computationally efficient in all investigated problems, and (2) the framework generalizes well to previously un-encoded environments.**

I. INTRODUCTION

Recently, machine learning has been infused in many solutions to notoriously difficult problems in robotics. These strategies have seen a wide array of success in visual sensing [10], task learning [21], and human-robot cooperation [26], among others fields. It is important to understand the potential impact of machine learning in all levels of a robot's computation stack including common sub-components to solving these problems, e.g., subroutines for solving the motion planning problem.

In this work, we apply machine learning to predict the validity of robot configurations, a common procedure in sampling-based motion planners. Sampling-based approaches have seen broad applicability to many high-dimensional and complex motion planning problems in robotics [7], computer graphics [17], and computational biology [9]. Most of these approaches rely on classification of robotic configurations into valid, e.g., collision-free, and invalid samples, while they work to construct graph approximations of a state space, called a roadmap. Thus, these approaches avoid full representation of the topology of the state space. However, as the dimensionality of the robot and the complexity of the workspace increase, the primitive operations of sampling-based methods become computationally obstructive [2].

*This work was not supported by any organization.

[1]Tuan Tran and Chinwe Ekenna are with the Department of Computer Science, University at Albany, SUNY, NY 12206, USA {ttran3, cekenna}@albany.edu.

[2]Jory Denny is with the Department of Mathematics and Computer Science, University of Richmond, VA 23173, USA jdenny@richmond.edu.

Prior research has focused on improved geometric analysis techniques for collision checking [20], approximately modeling state space obstacles to predict collision status of robot configurations, and lazily invoking the configuration validation subroutine [7], among others. While these methods have shown success in many applications, none have fully and permanently bounded the computational cost of validating configurations in sampling-based motion planners.

We propose a novel framework to efficiently and effectively predict the validity of robot configurations in complex motion planning problems. Figure 1 shows the two parts of our framework: (1) a Contractive AutoEncoder (CAE) [24], which embeds the robot's workspace into a latent space, and (2) a feed-forward neural network, which predicts the validity of a robots' configuration given the CAE encoding of the workspace. We examine the potential impact of such an approach in a variety of high dimensional robots in complex 2D and 3D workspaces. Our research shows that:

1) Our framework is computationally efficient in all investigated problems, and
2) the overall approach generalizes well to previously un-encoded environments.

As such, our framework can be applied to many existing sampling-based routines to improve their computational efficiency.

The rest of this paper is organized as follows: Section 2 provides some of the related work for this problem. In section 3, we introduce notation and state our problem definition. Section 4 presents our proposed framework. Section 5 details the framework architecture and implementation. Section 6 presents the results. In Section 7 draws conclusions and discusses directions for future work.

II. RELATED WORK

The main paradigms of sampling-based motion planning are the Probabilistic RoadMap (PRM) [16] and the Rapidly-exploring Random Tree (RRT) algorithms [18]. Both PRM and RRT are known to be probabilistically complete, i.e., if a (robust) solution exists, then a solution will almost surely be found as the number of samples increases [2]. Many proposed variations on PRM and RRT improve their performance for a variety of scenarios. Inoue et.al. [15] proposed a robot path planning method that combines RRT and long short-term memory network, which recalls the path of a robot by training with a large number of paths generated by RRT. Zhang et.al. [29] improved RRT algorithm by using target bias sampling strategy, considering both distance and rotation angle when choosing the nearest neighbor, and making the planned path as smooth as possible

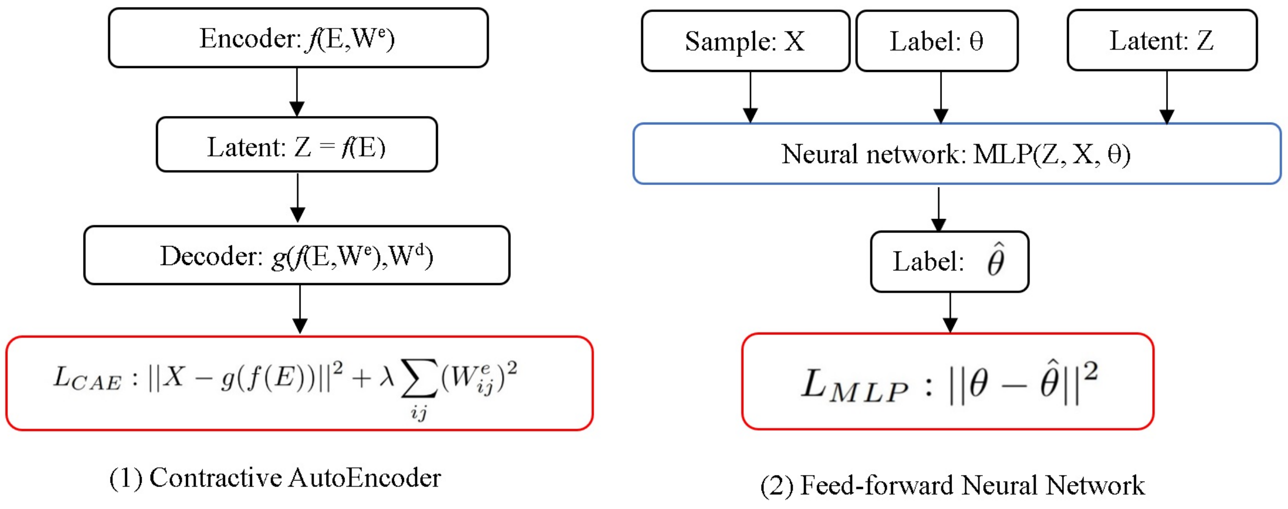
Encoder: f(E,W^e)
Latent: Z = f(E)
Decoder: g(f(E,W^e),W^d)
$L_{CAE} : ||X - g(f(E))||^2 + \lambda \sum_{ij} (W_{ij}^e)^2$
(1) Contractive AutoEncoder
Sample: X
Label: θ
Latent: Z
Neural network: MLP(Z, X, θ)
Label: $\hat{\theta}$
$L_{MLP} : ||\theta - \hat{\theta}||^2$
(2) Feed-forward Neural Network

Fig. 1. Proposed Framework. The red boxes indicate the training objectives. The blue box is used for online path planning problem.

using radial basic function neural network. Hauser proposed Lazy-PRM* [11] a lazy collision checking strategy, which avoids checking connections that have no chance to improve upon the current best path, thus reducing the per-sample computational cost and accelerating solution convergence. This work likewise aims to improve the performance of sampling-based approaches — specifically through reducing the cost of collision detection routines.

Representation learning [1] mainly aims to extract features from unstructured data, to either achieve a lower dimensional representation (often referred to as encoding) or learn features for supervised learning or reinforcement learning. There have been some recent works to utilize a learnt low-dimensional latent model for motion planning. Ichter et al. [13] and Zhang et al [28] used the conditional variational autoencoders to learn a non-uniform sampling methodology of a sampling-based motion planning algorithm. Chen et al. [4] proposed a time-dependent variational autoencoder to learn dynamic motion primitives for humanoids, which could be applied to sampling configurations in motion planning. Qureshi et al. [23] presented a neural network-based planning algorithm which learns a low-dimensional representation of the obstacle space by first using a Contractive Autoencoder to encode a point cloud of the obstacles into a latent space and then using a feed-forward neural network to predict the next step an optimal planner would take given a start and goal. In this work, we aim to use a lower dimensional representation to reduce the cost of collision detection for use in motion planning algorithms.

The neural network is becoming prominent in path planning for its outstanding non-linear mapping ability. The shortcomings of neural network models include: hardware requirements to reduce training time and difficulty of obtaining the best parameters for learning and difficult to obtain the best parameters. However, with the fast development of hardware computing capabilities and undemanding requirements of path planning, those shortcomings are fading steadily. Li et al. [19] addressed motion planning problem for vehicles with non-linear dynamics in a clustered environment using near-optimal RRT, which utilizes neural network to estimate the cost function considering non-linear kinodynamic constraints. Moreover, Shi el at. [25] presented a motion planning system based on hybrid deep learning, which uses a convolutional neural network to reduce the dimension of the input image, a recurrent neural networks to create a path tracking model, and a fully connected neural network to construct a control model. Chen el at. [5] proposed an optimal robot path planning system that builds a map, plans optimal paths, and maneuvers mobile robots, in which a simplified neural network is used to calculate the optimal trajectory for the robot. In this work, we make use of the feed-forward neural network to quickly and efficiently determine the validity of generated samples for the motion planning problem.

III. PROBLEM STATEMENT

In this section, we define the problem addressed and denote various mathematical notations used throughout this work.

A. Motion Planning

Let $X \subset \mathbb{R}^d$ denote a given state space, where $d \in \mathbb{N}$ is the dimensionality of the state space. The obstacle and obstacle-free state spaces are defined as $X_{obs} \subset X$ and $X_{free} = X \backslash X_{obs}$ respectively. A path τ for the robot is defined as a series of states $\{x_0, ..., x_n\}$, where n is the number of time steps in the path. A path is said to be feasible if it lies entirely in the obstacle-free space X_{free}

The motion planning problem is defined as: given a state space $X = X_{free} \cup X_{obs}$, an initial state $x_{init} \in X_{free}$, and a goal region $X_{goal} \subseteq X_{free}$, find a feasible path $\tau \in X_{free}$ such that $\tau_{start} = x_{init}$ and $\tau_{end} \in X_{goal}$. If no such path exists, report failure.

B. Sampling-based Motion Planning

A common solution to the motion planning problem is the utilization of sampling-based methods. Most often, sampling-based motion planners randomly select and progressively expand and connect robot configurations to form approximate graph representations of X_{free}, called roadmaps. A roadmap encodes valid states as its nodes and transitions between them as its edges. In order to find a path, a starting state and goal region are connected to the roadmap, and a feasible path is extracted from it.

These techniques employ invocation of a "black box" collision detection module that classifies any state in either X_{free} or X_{obs} in order to avoid explicit construction of X_{obs}. This is used in verification of both the nodes and the edges of any roadmap constructed using these methodologies. Commonly sampling-based strategies offer guarantees on probabilistic completeness and asymptotic optimality [14].

IV. FRAMEWORK

This section introduces our proposed framework. It is a neural network based motion planner comprised of two phases. The first phase trains a Contractive AutoEncoder (CAE) and a feed-forward neural network in an offline, apriori fashion. The second phase performs path finding computations online using the trained CAE and feed-forward neural network.

A. Phase One: Training Phase

The proposed framework uses two neural network models to solve the motion planning problem. The first is a CAE that embeds a representation of the environment, corresponding to a point cloud, into a latent space. The second is a feed-forward neural network which predicts the validity of a given input sample and the CAE encoding of the environment. Thus, instead of using the information of the whole workspace to decide whether a sample is valid or invalid, only the limited information from the encoded latent space is used.

1) Contractive AutoEncoder (CAE): The standard construction of a CAE consists of training neural networks for the encoder and the decoder. The encoder and the decoder are trained together using the reconstruction error. The output of the encoder represents a reduced representation of the initial input, and the decoder reconstructs that initial input from an encoded representation by minimizing a cost or loss function.

A CAE is used to embed a workspace point cloud into an invariant and robust latent space $Z \in \mathbb{R}^m$, where $m \in \mathbb{N}$ is the dimensionality of the latent space. Let $f(x, W^e)$ be an encoding function with weight matrix W^e that encodes an input vector $x \in X$ into a vector in the latent space $z \in Z$. A decoding function $g(z, W^d)$, with weight matrix W^d, decodes a vector from the latent space $z \in Z$ back into a vector in the workspace $x \in X$. The objective function (loss function) for the CAE is:

$$L_{CAE} = \frac{1}{|D|} \sum_{x \in D} ||x - g(f(x))||^2 + \lambda \sum_{ij} (W^e_{ij})^2 \quad (1)$$

where λ is a penalizing coefficient and D is the point cloud data from $\mathbb{N}$ different workspaces. The penalizing term forces the feature space $f(x, W^e)$ to be contractive in the neighborhood of the training data which results in an invariant and robust feature learning [24].

2) Multi-layer Perceptrons (MLP): We use a feed-forward neural network to perform sample validity. Given a workspace encoding $f(x, W^e) \in Z$, the samples' information, MLP predicts whether a sample is valid or not. The training objective for the MLP is to minimize the mean-squared-error (MSE) loss between the predicted sample's label and the its actual label.

B. Online Path planning

The online phase exploits the neural models from the offline phase to do motion planning in cluttered and complex environments. Algorithm 1 presents the overall path generation procedure. First, a set of samples S is generated based on specification of a robot R and its particular workspace W. Then, using the offline-trained CAE, W is encoded into a latent space Z. Next, the offline-trained MLP takes the sample set S and the encoded latent space Z as input to determine a set of probably valid samples $S' \in S$. Then, the probably valid sample set S' is connected to create a roadmap by a proximity search function. Finally, from that obtained roadmap, a local planning function would return a feasible trajectory between a given start and goal. We will discuss in more detail for each step in our algorithm in the following sections.

Algorithm 1 SBMP Variant

```
Input: New environment $D$ (point cloud data), Query $x_{init}$, $X_{goal}$
1: Randomly sample a set of states, $S$.
2: Use the CAE to encode $D$ into the latent space $Z$.
3: Feed $S$ and $Z$ into MLP, which predicts the validity of each sample in $S$. After this step $S'$ a set of most probably valid samples are retained.
4: Feed $S'$ into a standard PRM approach to yield a roadmap $R$.
5: Extract a path $\tau$ from $R$ between $x_{init}$ and $X_{goal}$.
```

1) Sample Generation: We use the following samplers to generate the initial sample set S from each workspace: Basic PRM (BS) [16], Obstacle Based PRM (OB) [16], Gaussian Sampler (G) [3], and Bridge Test (BT) [12].

2) Environment Encoder: The encode function $f(x, W^e)$, trained in the offline phase, is used to encode the environment point cloud $x \in X$ into a latent space $Z \in \mathbb{R}^m$.

3) Sample's Validity: The MLP is a feed-forward neural network from the offline phase which takes the workspace encoding Z, the samples' information S, and predict samples' validity to create S'. To introduce stochasticity into the MLP and to prevent over-fitting, some of the hidden units in each hidden layer of the MLP were dropped out with a probability p. Dropout is applied layer-wise to a neural network and it drops each unit in the hidden layer with a probability $p : [0, 1] \in \mathbb{R}$.

4) Connection and Path Planning: We use straight line local planner. Particularly a PRM approach is applied to S' to generate a roadmap R, and path is extracted and verified.

V. IMPLEMENTATION DETAILS

This section gives the implementation details of our framework. The proposed neural models, CAE and MLP, are implemented in PyTorch [22].

A. Data Collection

1) Workspace Data: To generate different environments for training and testing, a number of quadrilateral blocks were placed in the operating region of 31×31 for 2D simple workspaces (2DS), $41 \times 41 \times 6$ for 3D office-like workspaces (3DO), and $11 \times 11 \times 11$ for 3D clutter workspaces (3DC). The placement of these blocks were randomly chosen in the operating region. Each random placement of the obstacle blocks leads to a different workspace. Currently, we do not consider any rotation of the blocks, i.e., they are axis aligned.

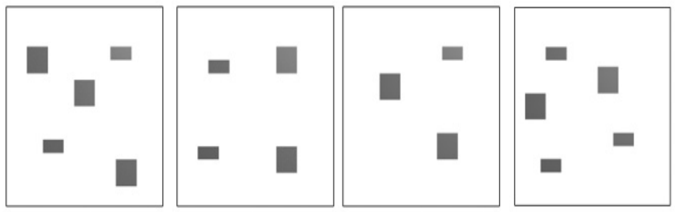
Fig. 2. Examples of 2D simple workspace

We represent the workspaces as a point cloud with values of -1 and 1, in which 1 means there is a obstacle at that point, and -1 means the point is in free space. The input to the encoder is a point cloud of size $n \times m$ for 2D and $n \times m \times k$ for 3D, where n, m, and k are the number of points along each dimension of the environment.

We use multiple workspace representations to train our CAE. The training data set of the 2D workspaces comprised of 30 environments, and for testing, two types of test data sets were created to evaluate the proposed method. The first test data set comprised of the 30 workspaces used in training, and the second test data set comprised of 10 previously unseen workspaces, i.e., 10 workspaces not from the training set. There are around 3 to 5 obstacle blocks with different shapes and sizes in each workspace.

The training data set of the 3D office-like workspaces comprised of 100 environments, and for testing 100 known workspaces and 20 previously unseen workspaces with 25 to 30 obstacle blocks in each workspace were used.

The training data set of the 3D clutter workspaces comprised of 50 environments, and for testing 30 known workspaces and 10 previously unseen workspaces with 110 to 125 same shape and size obstacle blocks in each workspace were used.

To test the effectiveness of our proposed framework, we incorporate it into the standard PRM approach using

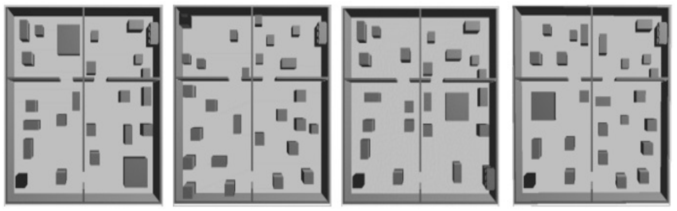
Fig. 3. Examples of 3D office-like workspace

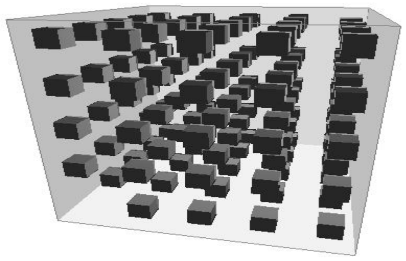
Fig. 4. Example of 3D clutter workspace

algorithm 1 to predict a sample's validity. However, instead of having a large set of samples S', we provide a small set of samples S' and let the algorithm decide whether those supplied samples are enough or there is need to generate more samples. We think that providing the algorithm with too many samples would negatively impacts the performance of neighborhood finding routines in sampling-based planners.

For 2D simple and 3D office-like environment, we randomly generate 10 different environments with 10 different start and goal positions for each of them; and 100 3D clutter environments with 1 different start and goal positions for each of them to test the performance of our proposed framework in terms of time.

2) Testing Data: For the 2D simple environments, we use a point-mass robot with 2 degrees of freedom (DoF) that represents its $\{x, y\}$ position. For the 3D office-like environments, we use a complex robot, specifically KuKa robot [6], which has 5 joints, with total of 8 DoF. The 8 DoFs represent the robot's $\{x, y\}$ position, rotation, and each joint's rotation. For the 3D clutter workspaces, we use snake-like robots with 7 DoF and 9 DoF. The DoFs represent the robot's $\{x, y, z\}$ position, $\{\alpha, \beta, \gamma\}$ rotation, and each joint's rotation.

We generate 100 samples for each 2D workspace and 200 samples for each 3D workspace for training and testing our MLP. Those samples are approximately distributed 50/50 between valid and invalid samples.

To test the performance of our framework, we generate 200 samples regardless of their validity for each 2D simple workspace, 2000 samples for each 3D office-like workspace, and 400 samples for each 3D clutter workspace.

B. Model Architecture

1) Contractive AutoEncoder: Since the structure of the decoders is the inverse of the encoder, we only describe the structure of the encoder.

For the 2D workspaces, the encoding function $f(x, W^e)$ and decoding function $g(f(x, W^e), W^d)$ consist of five linear layers and one output layer, where each layer uses Parametric Rectified Linear Unit (PReLU) [27] as its activation function, for loss function, we use mean square error with weight-decay. The layers 1 to 5 transforms the input vector 31×31 to 512, 256, 128, 64, and 32 hidden units, respectively. The output layer takes 32 units as input and outputs 12 units. Hence the environment representation in the latent space is a vector of size 12. Then, those 12 units are used by the decoder to reconstruct the 31×31 space.

For the 3D office-like workspaces, the encoding function $f(x)$ and decoding function $g(f(x))$ are the same as in 2D simple workspace, but consist of seven linear layers and one output layer. Layers one to seven transform the input vectors to 5043, 3125, 1600, 800, 400, 200, 100 hidden units, respectively. The output layer takes the 100 units from the seventh layer and transform them to 50 units, which represent the environment in latent space.

For the 3D clutter workspaces, layers one to six transform the input vectors to 1000, 800, 600, 400, 200, 100 hidden units, the output layer transforms the input 100 units to output 50 units.

2) Multi-layer Perceptrons: The input is given by concatenating the encoded workspace's representation Z, and the DoF for each robot from a given state. Each of layers is a sandwich of a linear layer, Parametric Rectified Linear Unit (PReLU), and Dropout (p).

For the 2D workspaces, MLP is a 4-layer neural network. Layers one and two map the input vectors to 6, 4 units, respectively. The output layer takes the 4 units and transform them to the validity of each sample, which is either valid or invalid.

For the 3D workspaces, MLP is a 7-layer neural network. Layers one to six map the input vectors to 50, 40, 30, 20, 10, 5 units, respectively. The output layer takes the 5 units from the sixth layer and transform them to the validity of each sample.

3) Parameters: To train the neural models CAE and MLP for both 2D and 3D workspace, the Adagrad [8] optimizer was used with the learning rate of 0.1. The Dropout probability p and penalizing term λ were set to 0.5 and 0.001, respectively.

VI. RESULTS

This section presents the results of framework for the motion planning of a point-mass robot in the 2D environments and the KuKa robot and a snake-like robot in 3D environments.

	Seen	Unseen
2DS	100%	100%
3DO	96%	94%
3DC	98%	95%

TABLE I
PERFORMANCE OF CAE

Table I shows the recall for the CAE. For 2DS, the average accuracy of our CAE is 100% with the variance of 0 for both already seen workspaces and unseen workspaces. For 3DO, the average accuracy of our CAE is 96% with the variance of 3% for already seen workspaces and the average accuracy is 94% with variance of 5% for unseen workspaces. The accuracy and variance are 98% and 1% and 95% and 3% for seen and unseen 3D clutter, respectively. Overall, the accuracy for our CAE is excellent. Thus, we are able to learn the underlying structure of the workspaces quite well.

	Seen	Unseen
2DS	92%	87%
3DO	74%	70%
3DC-7	74%	71%
3DC-9	72%	70%

TABLE II
PERFORMANCE OF MLP

Table II shows the accuracy for the MLP. For 2D workspaces, the average accuracy of our MLP is 92% with the variance of 2% for already seen workspaces and the average accuracy is 87% with variance of 5% for unseen workspaces. For 3DO workspace, the average accuracy of our MLP is 74% with the variance of 7% for already seen workspaces, and the average accuracy is 71% with variance of 10% for unseen workspaces. The average accuracy is 72% with 5% variance (seen), and 70% with 9% variance (unseen) for 3DC. Overall, the accuracy of our MLP is acceptable. We think the MLP has low accuracy because the same sample could be valid for most of the workspaces but invalid in a small set of workspaces, or vice versa, thus the distinctions between valid and invalid samples are not captured properly.

	BS	BSF	OB	OBF	G	GF	BT	BTF
2DS	0.12	0.10	0.38	0.31	0.09	0.09	0.13	0.13
3DO	40.9	41.6	43.7	39.9	42.3	39.4	40.4	44.4
3DC-7	0.53	0.56	0.65	0.55	0.26	0.29	0.95	0.44
3DC-9	0.73	0.51	3.27	2.4	3.9	3.3	7.1	4.5

TABLE III
TIME COMPARISON WHEN APPLYING OUR PROPOSED FRAMEWORK.

Table III shows the execution time comparison in seconds when applying the new framework with Basic PRM (BSF), Obstacle Based PRM (OBF), Gaussian (GF), and Bridge Test (BTF) versus not applying the new framework when using BS, OB, G, and BT for solving motion planning problems. In 2D workspaces, the average execution time for each query for each scenario is the following: UR: 0.12 ± 0.01, URF: 0.10 ± 0.01, OB: 0.38 ± 0.05, OBF: 0.31 ± 0.03, G: 0.09 ± 0.01, GF: 0.09 ± 0.01, BT: 0.13 ± 0.01, BTF: 0.13 ± 0.01. Notable, our framework has comparable performance. Additionally, we notice there is a nearly 25% improvement when applying our framework to OB.

Figure 5 shows the performance of our proposed framework in 3DO workspace. There are improvements when using our framework with Obstacle Based (9%) and Gaussian (7%) sampler. However, there is a slow-down for the Uniform Random (2%) and Bridge Test (10%) samplers. We

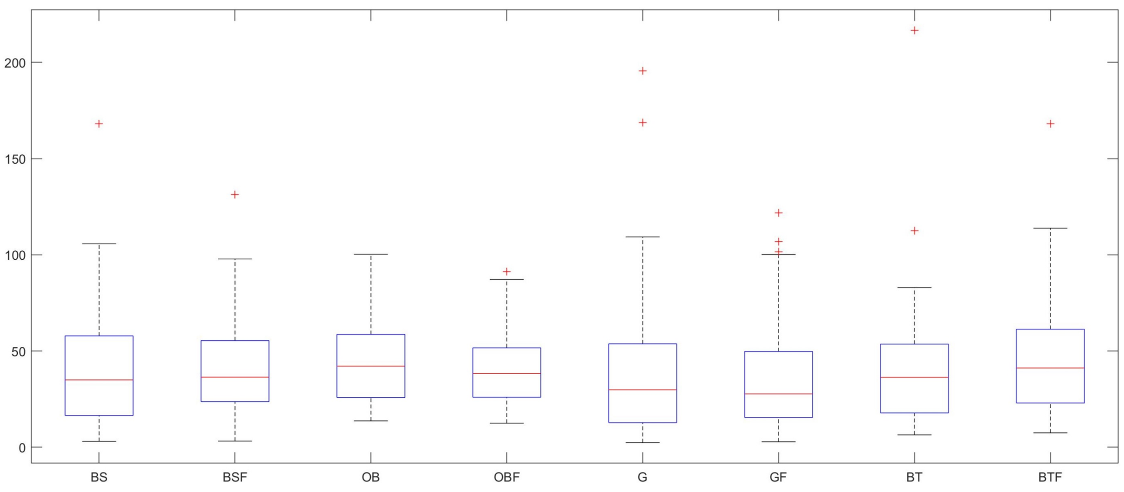

Fig. 5. Performance of 3DO

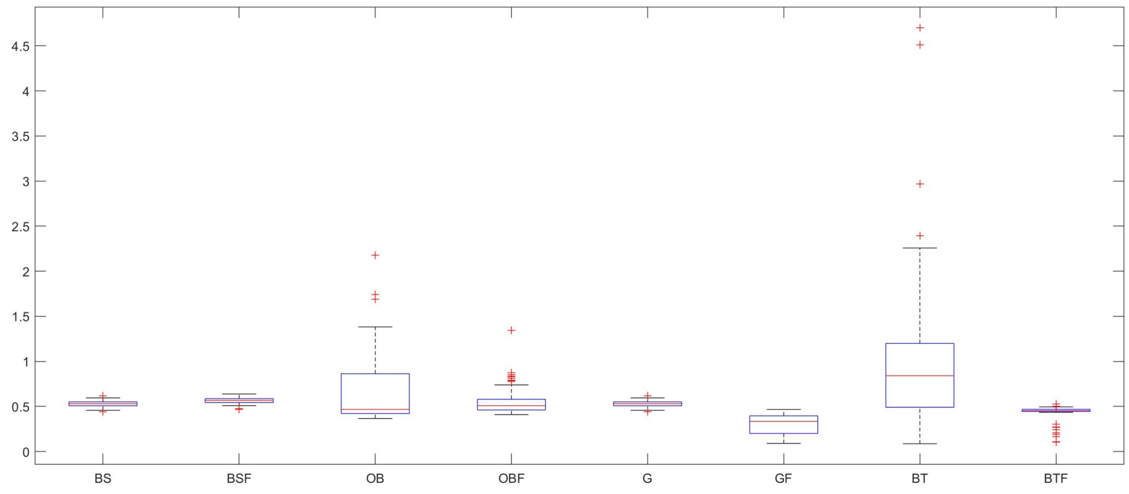

Fig. 6. Performance of 3DC-7

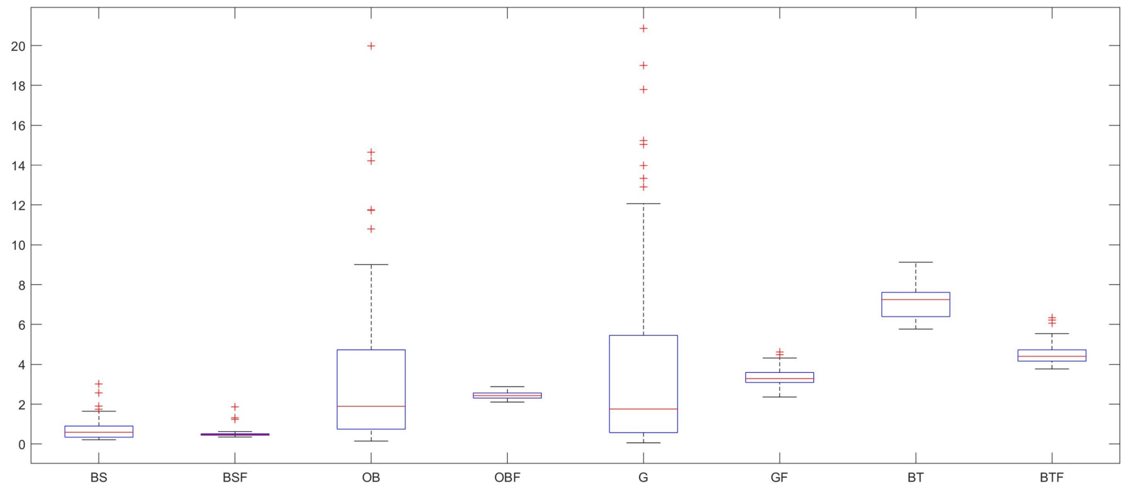

Fig. 7. Performance of 3DC-9

suspect that it could be that the supplied samples are not good enough, thus there is the need for re-sampling.

Figure 6 shows the performance of our proposed framework in 3DC workspace with a 7-DoF robot. The BT samplers perform the worst because of the open nature of our testing environment.

Figure 7 shows the performance of our proposed framework in 3DC workspace with a 9-DoF robot. There are quite notable time improvements when using our framework to augment the motion planning algorithms.

Overall, the performance of our framework is quite significant. Moreover, the standard deviations in our framework decreased compared to not using our framework. Thus, when applying our framework, the algorithms become more stable in solving queries reliably.

VII. Conclusion

In this paper, we present a fast and efficient neural network framework for sampling-based motion planners. The framework consists of a Contractive AutoEncoder that encodes a point cloud representation of a robot's environment into a latent feature space and a feed-forward neural network that takes the environment encoding and robot configuration details to predict the validity of that configuration. The main advantage of our framework is that we use an encoded environment and neural network to reduce or even remove expensive collision detection operations.

In the future, we would like to increase the accuracy of our MLP by pre-processing a configuration's information before training. Since the same sample could be valid for most of the workspaces but invalid in a small set of workspaces, we want to pre-process to highlight this characteristic of valid and invalid samples. Also, we want to apply our framework for connection routines by using another MLP to check for valid and invalid connections. Moreover, we also want to address the problem with rotated obstacles by having a finer representation of each workspace.

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli. Efficient collision checking in sampling-based motion planning. In *Algorithmic Foundations of Robotics X*, pages 365–380. Springer, 2013.

[3] V. Boor, M. H. Overmars, and A. F. Van Der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *ICRA*, pages 1018–1023, 1999.

[4] N. Chen, M. Karl, and P. van der Smagt. Dynamic movement primitives in latent space of time-dependent variational autoencoders. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 629–636. IEEE, 2016.

[5] Y.-W. Chen and W.-Y. Chiu. Optimal robot path planning system by using a neural network-based approach. In *2015 international automatic control conference (CACS)*, pages 85–90. IEEE, 2015.

[6] K. R. Corporation. Kuka youbot. www.youbot-store.com. Accessed: June 1, 2013.

[7] J. Denny, K. Shi, and N. M. Amato. Lazy toggle prm: A single-query approach to motion planning. In *2013 IEEE International Conference on Robotics and Automation*, pages 2407–2414. IEEE, 2013.

[8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[9] C. Ekenna, S. Thomas, and N. M. Amato. Adaptive local learning in sampling based motion planning for protein folding. *BMC systems biology*, 10(2):49, 2016.

[10] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[11] K. Hauser. Lazy collision checking in asymptotically-optimal motion planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2951–2957. IEEE, 2015.

[12] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, volume 3, pages 4420–4426. IEEE, 2003.

[13] B. Ichter, J. Harrison, and M. Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094. IEEE, 2018.

[14] B. Ichter and M. Pavone. Robot motion planning in learned latent spaces. *arXiv preprint arXiv:1807.10366*, 2018.

[15] M. Inoue, T. Yamashita, and T. Nishida. Robot path planning by lstm network under changing environment. In *Advances in Computer Communication and Computational Sciences*, pages 317–329. Springer, 2019.

[16] L. Kavraki, P. Svestka, and M. H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, volume 1994. Unknown Publisher, 1994.

[17] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[18] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[19] Y. Li, R. Cui, Z. Li, and D. Xu. Neural network approximation based near-optimal motion planning with kinodynamic constraints using rrt. *IEEE Transactions on Industrial Electronics*, 65(11):8718–8729, 2018.

[20] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA conference on mathematics of surfaces*, volume 1, pages 602–608, 1998.

[21] C. L. Nehaniv and K. E. Dautenhahn. *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions.* Cambridge University Press, 2007.

[22] 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. 2017.

[23] A. H. Qureshi, M. J. Bency, and M. C. Yip. Motion planning networks. *arXiv preprint arXiv:1806.05767*, 2018.

[24] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011.

[25] C. Shi, X. Lan, and Y. Wang. Motion planning for unmanned vehicle based on hybrid deep learning. In *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pages 473–478. IEEE, 2017.

[26] P. Trautman, J. Ma, R. M. Murray, and A. Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[27] L. Trottier, P. Gigu, B. Chaib-draa, et al. Parametric exponential linear unit for deep convolutional neural networks. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–214. IEEE, 2017.

[28] C. Zhang, J. Huh, and D. D. Lee. Learning implicit sampling distributions for motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3654–3661. IEEE, 2018.

[29] P. Zhang, C. Xiong, W. Li, X. Du, and C. Zhao. Path planning for mobile robot based on modified rapidly exploring random tree method and neural network. *International Journal of Advanced Robotic Systems*, 15(3):1729881418784221, 2018.