Active Continual Learning for Planning and Navigation

Ahmed H. Qureshi, Yinglong Miao, Michael C. Yip
University of California San Diego
La Jolla, CA 92093, USA
{a1qureshi, y2miao, yip}@ucsd.edu

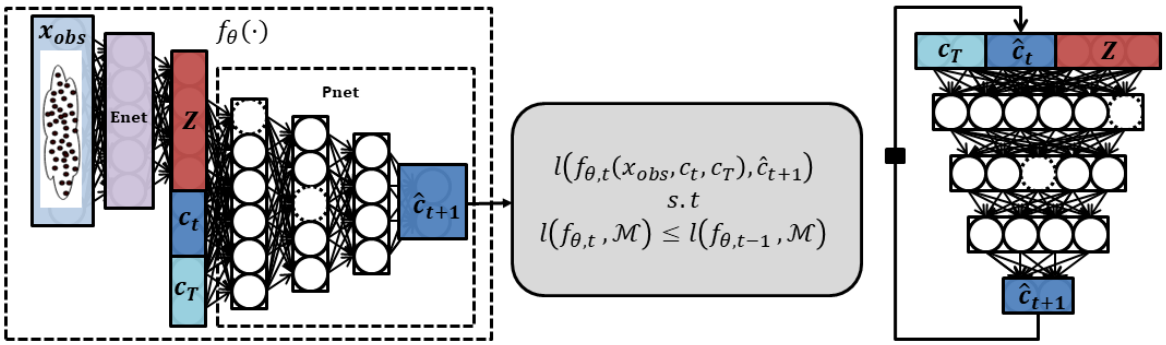
Abstract

Recent developments have led to imitation-based planners that learn by imitating expert demonstrations to solve general motion planning and navigation problems. These planners are known for their breakneck computational speed during online planning. However, training these methods offline requires a large number of expert demonstrations, which makes them impractical in cases where data is expensive to make and can come in streams on a need basis. For instance, in semi-autonomous driving, the demonstrations could only be provided on request for given planning problems. To address that challenge, we present an active continual learning approach that enables learning-based motion planners to learn from streaming data and actively ask for expert demonstrations when needed, drastically reducing the data required for training. Our results indicate that the proposed method consumes about 80 % lesser data than traditional approaches while exhibiting comparable planning performances.

Keywords: Continual Learning, Active Learning, Deep Learning, Planning & Navigation

1. Introduction

Robotic motion planning aims to compute a collision-free path for the given start and goal configurations (LaValle, 2006). As motion planning algorithms are necessary for solving a variety of complicated, high-dimensional problems ranging from autonomous driving (Lozano-Perez, 2012) to space exploration (Volpe, 2003), there arises a critical, unmet need for computationally tractable, real-time algorithms. The quest for developing computationally efficient motion planning methods has led to the development imitation-based planners that learn to plan by imitating an oracle planner (Bency et al., 2019; Ichter et al., 2018; Qureshi and Yip, 2018; Qureshi et al., 2019, 2020). These planners are known for their extremely fast computational speed during online planning. Some of these planners even provide theoretical completeness guarantees derived from an underlying classical planning method. For instance, Motion Planning Networks (MPNet), a deep neural network-based method, generates collision-free paths through divide-and-conquer as it divides the problem into sub-problems and outsources them, in worst-case, to a classical planner while retaining its computational benefits. However, standard MPNet training assumes the availability of complete data for offline training and cannot learn from streaming data while avoiding catastrophic forgetting, which is a known problem for deep learning-based methods.
In this paper, we present an active continual learning approach to train MPNet. Our method incorporates MPNet into a continual learning process and asks for expert demonstrations



(a) (b)

Figure 1: MPNet consists of encoder network (Enet) and planning network (Pnet). Fig (a) shows that Pnet and Enet can be trained end-to-end and can learn under continual learning settings from streaming data using constraint optimization and episodic memory $\mathcal{M}$ for a given loss function $l(\cdot)$. Fig (b) shows the online execution of MPNet's neural models.

only when needed, hence improving the overall training data efficiency. This strategy is in response to practical and data-efficient learning where planning problems come in streams, and MPNet attempts to plan a motion for them. In case MPNet fails to find a path for a given problem, only then an Oracle Planner is called to provide an expert demonstration for learning. We compare our training method against: 1) standard offline batch learning which assumes the availability of all training data, and 2) continual learning with episodic memory which assumes that the expert demonstrations come in streams and the global training data distribution is unknown.

2. MPNet: A Neural Motion Planner

MPNet comprises of two neural networks: an encoder network (Enet) and a planning network (Pnet). Enet takes the robot's surrounding information such as a raw point-cloud and transforms it to a latent space embedding using a fully-connected deep neural network. Pnet takes the encoding of the environment, the robot's current state and goal state to output samples for a path generation. In remaining section, we describe the notations necessary to outline MPNet.

Let robot configuration space (C-space) be denoted as $\mathcal{C} \subset \mathbb{R}^d$ comprising of obstacle space $\mathcal{C}_{\text{obs}}$ and obstacle-free space $\mathcal{C}_{\text{free}} = \mathcal{C} \backslash \mathcal{C}_{\text{obs}}$, where d is the C-space dimensionality. Let robot's surrounding environment, also known as workspace, be denoted as $\mathcal{X} \subset \mathbb{R}^m$, where m is a workspace dimension. Like C-space, the workspace also comprise of obstacle, $\mathcal{X}_{\text{obs}}$, and obstacle-free, $\mathcal{X}_{\text{free}} = \mathcal{X} \backslash \mathcal{X}_{\text{obs}}$, regions. The workspaces could be up to 3-dimensions whereas the C-space can have higher dimensions depending on the robot's degree-of-freedom (DOF). Let robot initial and goal configuration space be $c_{\text{init}} \in \mathcal{C}_{\text{free}}$ and $c_{\text{goal}} \subset \mathcal{C}_{\text{free}}$, respectively. Let $\sigma = \{c_0, \cdots, c_T\}$ be an ordered list of length T. We assume σ_i corresponds to the i-th state in σ, where $i = [0, T]$. For instance σ_0 corresponds to state c_0. Furthermore, we consider σ_{end} corresponds to the last element of σ, i.e., $\sigma_{\text{end}} = c_T$.

We consider a practical scenario, where MPNet plans feasible, near-optimal paths using raw

point-cloud/voxel data of obstacles $x_{\text{obs}} \subset \mathcal{X}_{\text{obs}}$. However, like other planning algorithms, we do assume an availability of a collision-checker that verifies the feasibility of MPNet generated paths based on $\mathcal{X}_{\text{obs}}$. Precisely, Enet, with parameters θ^e, takes the environment information x_{obs} and compresses them into a latent space, i.e., $Z \leftarrow \text{Enet}(x_{\text{obs}}; \theta^e)$. Pnet, with parameters θ^p, takes the environment encoding Z, robot's current or initial configuration $c_t \in \mathcal{C}_{\text{free}}$, and goal configuration $c_{\text{goal}} \subset \mathcal{C}_{\text{free}}$ to produce a trajectory through incremental generation of states $\hat{c}_{t+1}$ (Fig. 1 (b)), $\hat{c}_{t+1} \leftarrow \text{Pnet}(Z, c_t, c_{\text{goal}})$.

3. Active Continual Learning

In this section, we formally present the active continual learning (ACL) approach for learning-based planning algorithms such as MPNet. MPNet primarily used offline batch learning method (Qureshi et al., 2019; Qureshi and Yip, 2018) which requires the availability of complete data to train MPNet offline before running it online to plan motions for unknown/new planning problems. The ACL incorporates MPNet into the continual learning process where MPNet actively asks for an expert demonstration when needed for the given problem.

In ACL settings, the data comes in streams with targets, i.e.,

$$(s_1, y_1, \cdots, s_i, y_i, \cdots, s_N, y_N)$$

where $s = (c_t, c_T, x_{\text{obs}})$ is the input to MPNet comprising of the robot's current state c_t, the goal state c_T, and obstacles information x_{obs}. The target y is the next state c_{t+1} in the expert trajectory given by an oracle planner. Generally, continual learning using neural networks suffers from the issue of catastrophic forgetting since taking a gradient step on a new datum could erase the previous learning.

To overcome catastrophic forgetting, we employ the Gradient Episodic Memory (GEM) method for lifelong learning (Lopez-Paz and Ranzato, 2017). GEM uses the episodic memory $\mathcal{M}$ that has a finite set of continuum data seen in the past to ensure that the model update doesn't lead to negative backward transfer while allowing only the positive backward transfer. For MPNet, we adapt GEM for the regression problem using the following optimization objective function.

$$\min_{\theta} l(f_\theta^t(s), y) \text{ s.t } \hat{\mathbb{E}}_{(s,y)\sim\mathcal{M}}[l(f_\theta^t(s), y)] \leq \hat{\mathbb{E}}_{(s,y)\sim\mathcal{M}}[l(f_\theta^{t-1}(s), y)] \tag{1}$$

where $l = \|f_\theta^t(s) - y\|^2$ is a squared-error loss, f_θ^t is the MPNet model at time step t (see Fig. 1). Furthermore, note that, if the angle between proposed gradient (g) at time t, and the gradient over $\mathcal{M}$ ($g_{\mathcal{M}}$) is positive, i.e., $\langle g, g_{\mathcal{M}} \rangle \geq 0$, there is no need to maintain the old function parameters f_θ^{t-1} because the above equation can be formulated as:

$$\langle g, g_{\mathcal{M}} \rangle := \left\langle \bigtriangledown_\theta \, l\big(f_\theta(s), y\big), \hat{\mathbb{E}}_{(s,y)\sim\mathcal{M}} \bigtriangledown_\theta \, l\big(f_\theta(s), y\big) \right\rangle \tag{2}$$

where $\hat{\mathbb{E}}$ denotes empirical mean.

In most cases, the proposed gradient g violates the constraint $\langle g, g_{\mathcal{M}} \rangle \geq 0$, i.e., the proposed gradient update g will cause increase in the loss over previous data. To avoid such violations,

Lopez-Paz and Ranzato (2017) proposed to project the gradient g to the nearest gradient g' that keeps $\langle g', g_{\mathcal{M}} \rangle \geq 0$, i.e,

$$\min_{g'} \frac{1}{2} \|g - g'\|_2^2 \text{ s.t } \langle g', g_{\mathcal{M}} \rangle \geq 0 \tag{3}$$

The projection of proposed gradient g to g' can be solved efficiently using Quadratic Programming (QP) based on the duality principle, for details refer to (Lopez-Paz and Ranzato, 2017).

Various data parsing methods are available to update the episodic memory $\mathcal{M}$. These sample selection strategies for episodic memory play a vital role in the performance of continual/life-long learning methods such as GEM (Isele and Cosgun, 2018). There exist several selection metrics such as surprise, reward, coverage maximization, and global distribution matching (Isele and Cosgun, 2018). In our ACL framework, we use a global distribution matching method to select samples for the episodic memory. For details and comparison of different sample selection approaches, please refer to (Qureshi et al., 2020). The global distribution matching method, also known as reservoir sampling, uses random sampling techniques to populate the episodic memory. The aim is to approximately capture the global distribution of the training dataset since it is unknown in advance.

In addition to episodic memory $\mathcal{M}$, we also maintain a replay/rehearsal memory $\mathcal{B}^*$. The replay buffer lets MPNet rehearse by learning again on the old samples. We found that rehearsals further mitigate the problem of catastrophic forgetting and leads to better performance, as also reported by Rolnick et al. (2018) in reinforcement learning setting. Note that replay or rehearsal on the past data is done with the interval of replay period $r \in \mathbb{N}_{\geq 0}$.

Algorithm 1 presents the outline of ACL method. At every time step t, the environment generates a planning problem $(c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$ comprising of the robot's initial c_{init} and goal c_{goal} configurations and the obstacles' information $\mathcal{X}_{\text{obs}}$ (Line 7). Before asking MPNet to plan a motion for a given problem, we let it learn from the expert demonstrations for up to $N_c \in \mathbb{N}_{\geq 0}$ iterations (Line 9-10). If MPNet is not called or failed to determine a solution, an expert-planner is executed to determine a path solution σ for a given planning problem (Line 13). The expert trajectory σ is stored into a replay buffer $\mathcal{B}^*$ and an episodic $\mathcal{M}$ memory based on their sample selection strategies (Line 14-15). MPNet is trained (Line 16-19) on the given demonstration using the constraint optimization mentioned in Equations 1-3. Finally, similar to continual learning, we also perform rehearsals on the old samples (Line 20-25).

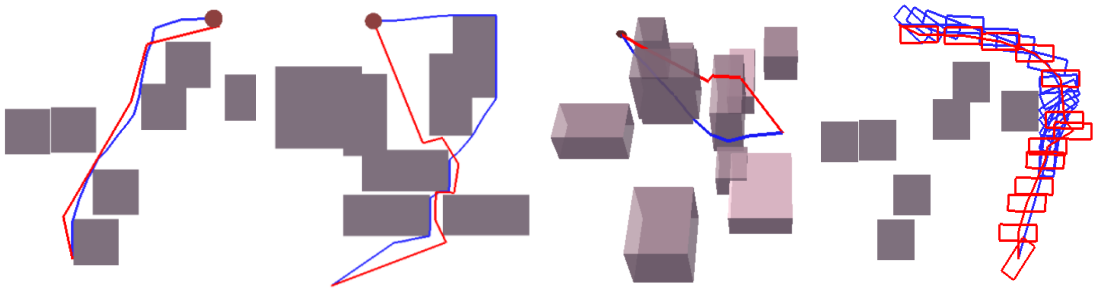
It can be seen that ACL introduces a two-level of sample selection strategy. First, ACL gathers the training data by actively asking for the demonstrations on problems where MPNet failed to find a path. Second, it employs a sample selection strategy that further prunes the expert demonstrations to fill episodic memory so that it approximates the global distribution from streaming data. The two-level of data selection in ACL improves the training samples efficiency while learning the generalized neural models for the MPNet.

4. Results & Conclusions

In this section, we present results evaluating MPNet models trained with batch learning (B), continual learning (C), and active continual learning (AC). We also compare these

Algorithm 1: Active Continual Learning

Initialize memories: episodic $\mathcal{M}$ and replay $\mathcal{B}^*$
Initialize MPNet f_θ with parameters θ.
Set the number of iterations C to pretrain MPNet.
Set the replay period r
Set the replay batch size N_B
for $t = 0$ ***to*** T **do**
  $\{\mathcal{X}_{\text{obs}}, c_{\text{init}}, c_{\text{goal}}\}_t \leftarrow$ GetPlanningProblem()
  $\sigma \leftarrow \varphi$ \\ an empty list to store path solution
  if $t > N_c$ **then**
    $x_{\text{obs}} \subset \mathcal{X}_{\text{obs}}$
    $\sigma \leftarrow f_\theta(x_{\text{obs}}, c_{\text{init}}, c_{\text{goal}})$
  if ***not*** σ **then**
    $\sigma \leftarrow$ GetExpertDemo$(\mathcal{X}_{\text{obs}}, c_{\text{init}}, c_{\text{goal}})$
    $\mathcal{B}^* \leftarrow \mathcal{B}^* \cup \sigma$
    $\mathcal{M} \leftarrow$ UpdateEpisodicMemory$(\sigma, \mathcal{M})$
    $g \leftarrow \hat{\mathbb{E}}_{(s,y)\sim\sigma} \bigtriangledown_\theta \, l\big(f_\theta(s), y\big)$
    $g_{\mathcal{M}} \leftarrow \hat{\mathbb{E}}_{(s,y)\sim\mathcal{M}} \bigtriangledown_\theta \, l\big(f_\theta(s), y\big)$
    Project g to g' using QP based on Equation 3
    Update parameters θ w.r.t. g'
  if $\mathcal{B}^*$.size() $> N_B$ ***and not*** t ***mod*** r **then**
    $\mathcal{B} \leftarrow$ SampleReplayBatch$(\mathcal{B}^*)$
    $g \leftarrow \hat{\mathbb{E}}_{(s,y)\sim\mathcal{B}} \bigtriangledown_\theta \, l\big(f_\theta(s), y\big)$
    $g_{\mathcal{M}} \leftarrow \hat{\mathbb{E}}_{(s,y)\sim\mathcal{M}} \bigtriangledown_\theta \, l\big(f_\theta(s), y\big)$
    Project g to g' using QP based on Equation 3
    Update parameters θ w.r.t. g'



(a) $t_R = 5.3, t_M = 0.02$ (b) $t_R = 6.8, t_M = 0.1$ (c) $t_R = 9.7, t_M = 0.1$ (d) $t_R = 27.8, t_M = 0.3$

Figure 2: Time comparison of MPNetPath (Red) and RRT* (Blue) for computing the near-optimal path solutions in example environments. Figs.(a) and (b) present simple 2D and complex 2D environments. Fig. (c) indicates complex 3D case whereas Figs. (d) shows the rigid-body-SE2 case.

models against state-of-the-art classical planners: RRT* (Karaman and Frazzoli, 2011), Informed-RRT* (Gammell et al., 2014), and BIT* (Gammell et al., 2015). For more details on the experimental setup, please refer to (Qureshi et al., 2020).

Methods	Environments			
	Simple 2D	Complex 2D	Complex 3D	Rigid-body-SE2
Informed-RRT*	1.10 ± 0.09	1.49 ± 0.16	2.76 ± 0.20	14.80 ± 2.83
BIT*	0.65 ± 0.30	1.61 ± 0.53	0.20 ± 0.04	6.52 ± 1.65
MPNetPath:NP (B)	0.02 ± 0.00	0.04 ± 0.01	0.07 ± 0.01	0.37 ± 0.02
MPNetPath:NP (C)	0.02 ± 0.00	0.05 ± 0.01	$0.08 \pm 0.0)$	0.39 ± 0.07
MPNetPath:NP (AC)	0.03 ± 0.01	0.06 ± 0.01	0.08 ± 0.01	0.42 ± 0.08

Table 1: Mean computation times with standard deviations are presented for MPNet (all variations), Informed-RRT* and BIT* on two test datasets, i.e., seen and unseen (shown inside brackets), in four different environments. In all cases, MPNet path planners trained with continual learning (C), active continual learning (AC) and offline batch learning (B) exhibt similar performance which is significantly better than classical planners such as Informed-RRT* and BIT* by an order of magnitude.

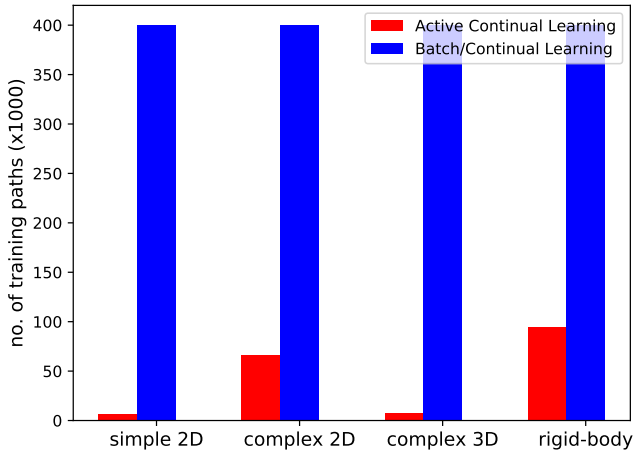


Figure 3: The number of training trajectories required by MPNet when trained with active continual learning as compared to traditional batch/continual learning approaches.

Fig. 2 shows the paths generated by MPNet(red), and its expert demonstrator RRT*(blue). The average computations times of MPNet and RRT* are denoted as t_R and t_M, respectively. It can be seen that MPNet finds paths of similar lengths as its expert demonstrator RRT* while retaining consistently low computational time. Overall, we observed that MPNet is at least $100\times$ faster than RRT* and finds paths that are within a 10% range of RRT*'s paths cost.

Table 1 compares mean computation times of different MPNet models with each other and against advanced classical planners in four different scenarios as shown in the Fig. 2. Fig. 3 shows the number of training paths consumed by all MPNet models presented in the Table 1. It can be seen that ACL exhibits significant improvement in data-efficiency and yet provides similar performance as conventional training methods in terms of computation times and success rates. Furthermore, the success rate of all MPNet models were compareable to each other, i.e., $90 - 99\%$.

References

M. J. Bency, A. H. Qureshi, and M. C. Yip. Neural path planning: Fixed time, near-optimal path generation via oracle imitation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3965–3972, 2019.

Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2997–3004. IEEE, 2014.

Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 3067–3074. IEEE, 2015.

Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094. IEEE, 2018.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017.

Tomás Lozano-Perez. *Autonomous robot vehicles*. Springer Science & Business Media, 2012.

Ahmed H Qureshi and Michael C Yip. Deeply informed neural sampling for robot motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6582–6588. IEEE, 2018.

Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019.

Ahmed H Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 2020.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682*, 2018.

Richard Volpe. Rover functional autonomy development for the mars mobile science laboratory. In *Proceedings of the 2003 IEEE Aerospace Conference*, volume 2, pages 643–652, 2003.