Path Planning using Neural A* Search

Ryo Yonetani [* 1] **Tatsunori Taniai** [* 1] **Mohammadamin Barekatain** [1 2] **Mai Nishimura** [1] **Asako Kanezaki** [3]

Abstract

We present *Neural A**, a novel data-driven search method for path planning problems. Despite the recent increasing attention to data-driven path planning, machine learning approaches to search-based planning are still challenging due to the discrete nature of search algorithms. In this work, we reformulate a canonical A* search algorithm to be differentiable and couple it with a convolutional encoder to form an end-to-end trainable neural network planner. Neural A* solves a path planning problem by encoding a problem instance to a guidance map and then performing the differentiable A* search with the guidance map. By learning to match the search results with ground-truth paths provided by experts, Neural A* can produce a path consistent with the ground truth accurately and efficiently. Our extensive experiments confirmed that Neural A* outperformed state-of-the-art data-driven planners in terms of the search optimality and efficiency trade-off. Furthermore, Neural A* successfully predicted realistic human trajectories by directly performing search-based planning on natural image inputs[1].

1. Introduction

Path planning refers to the problem of finding a valid low-cost path from start to goal points in an environmental map. *Search-based* planning, including the popular A* search (Hart et al., 1968), is a common approach to path planning problems and has been used in a wide range of applications such as autonomous vehicle navigation (Paden et al., 2016), robot arm manipulation (Smith et al., 2012), and game AI (Abd Algfoor et al., 2015). Compared to other planning approaches such as sampling-based plan-

*Equal contribution   [1]OMRON SINIC X, Tokyo, Japan [2]Now at DeepMind, London, UK. [3]Tokyo Institute of Technology, Tokyo, Japan. Correspondence to: Ryo Yonetani <ryo.yonetani@sinicx.com>.

[1]Project page: https://omron-sinicx.github.io/neural-astar/.
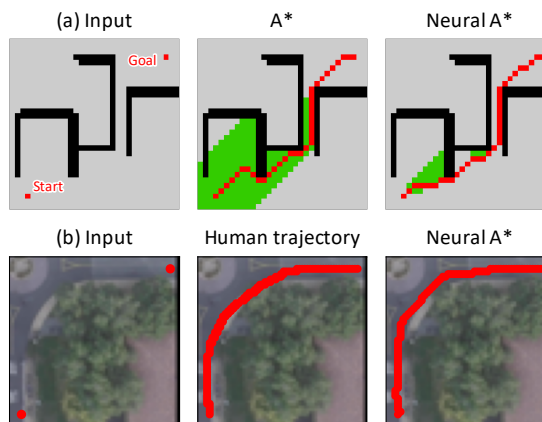
Figure 1. **Two Scenarios of Path Planning with Neural A*.** (a) Point-to-point shortest path search: finding a near-optimal path (red) with fewer node explorations (green) for an input map. (b) Path planning on raw image inputs: accurately predicting a human trajectory (red) on a natural image.

ning (González et al., 2015) and reactive planning (Tamar et al., 2016; Lee et al., 2018), search-based planning is guaranteed to find a solution path, if one exists, by incrementally and extensively exploring the map.

Learning how to plan from expert demonstrations is gathering attention as promising extensions to classical path planners. Recent work has demonstrated major advantages of such data-driven path planning in two scenarios: (1) finding near-optimal paths more efficiently than classical heuristic planners in point-to-point shortest path search problems (Choudhury et al., 2018; Qureshi et al., 2019; Takahashi et al., 2019; Chen et al., 2020; Ichter et al., 2020) and (2) enabling path planning on raw image inputs (Tamar et al., 2016; Lee et al., 2018; Ichter & Pavone, 2019; Vlastelica et al., 2020), which is hard for classical planners unless semantic pixel-wise labeling of the environment is given.

In this study, we address both of these separately studied scenarios in a principled fashion, as highlighted in Fig. 1. In contrast to most of the existing data-driven methods that extend either sampling-based or reactive planning, we pursue a search-based approach to data-driven planning with the intrinsic advantage of guaranteed planning success.

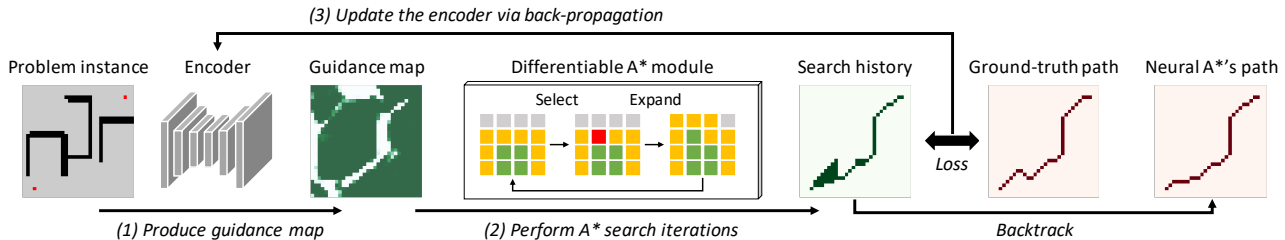Studies in this direction so far are largely limited due to

Figure 2. **Schematic Diagram of Neural A*.** (1) A path-planning problem instance is fed to the encoder to produce a guidance map. (2) The differentiable A* module performs a point-to-point shortest path search with the guidance map and outputs a search history and a resulting path. (3) A loss between the search history and the ground-truth path is back-propagated to train the encoder.

the difficulty arising from the discrete nature of incremental search steps in search-based planning, which makes the learning using back-propagation non-trivial. Some existing methods thus train heuristic cost functions at each grid point independently, which requires fine-grained expert annotations such as oracle planners running online (Choudhury et al., 2018) and exhaustive pre-computations of the optimal heuristic function of a planner (Takahashi et al., 2019). However, such rich annotations are not always available nor scalable, especially when involving labor-intensive manual annotation processes as in (Kim & Pineau, 2016; Kretzschmar et al., 2016; Pérez-Higueras et al., 2018). More recently, Vlastelica et al. (2020) have applied black-box optimization to combinatorial solvers integrated into neural networks, thus enabling end-to-end learning through combinatorial algorithms including search-based planning. This general-purpose optimization can be adapted to our problem. However, treating the entire search procedure as a black-box function loses the detailed track of internal search steps, making the training hard.

To address the non-differentiability of search-based planning, we propose a novel data-driven search-based planner named Neural A*. At its core, *we reformulate a canonical A* search algorithm to be differentiable* as a module referred to as the differentiable A*, by combining a discretized activation technique inspired by Hubara et al. (2016) with basic matrix operations. This module enables us to perform an A* search in the forward pass of a neural network and back-propagate losses through every search step to other trainable backbone modules. As illustrated in Fig. 2, Neural A* consists of the combination of a fully-convolutional encoder and the differentiable A* module, and is trained as follows: (1) Given a problem instance (*i.e.*, an environmental map annotated with start and goal points), the encoder transforms it into a scalar-valued map representation referred to as a guidance map; (2) The differentiable A* module then performs a search with the guidance map to output a search history and a resulting path; (3) The search history is compared against the ground-truth path of the input instance to derive a loss, which is back-propagated to train the encoder.

The role of the differentiable A* for training is simple: teaching the encoder to produce guidance maps that lead to minimizing the discrepancy between the resulting search histories and ground-truth paths. The encoder then learns to capture visual cues in the inputs that are effective for reproducing the ground-truth paths. This learning principle provides unified solutions to the aforementioned two problem scenarios. Specifically, in the shortest-path search scenario (Fig. 1a), where the ground-truth paths are given by optimal planners, the encoder is trained to find near-optimal paths efficiently by exploiting visual cues such as shapes of dead ends. Here, guidance maps are served to augment the input maps to prioritize which nodes to explore/avoid for improving the search optimality-efficiency trade-off. On the other hand, when the ground-truth paths are given by human annotators for raw image inputs (Fig. 1b), the encoder enables the planning directly on the images by learning passable and impassable regions from colors and textures as low- and high-cost nodes in guidance maps.

We extensively evaluate our approach on both synthetic (Bhardwaj et al., 2017) and real-world datasets (Sturtevant, 2012; Robicquet et al., 2016). Our results show that Neural A* outperformed state-of-the-art data-driven search-based planners (Choudhury et al., 2018; Vlastelica et al., 2020) in terms of the trade-off between search optimality and efficiency for point-to-point shortest path problems. Further, we demonstrate that Neural A* can learn to predict pedestrian trajectories from raw real-world surveillance images more accurately than Vlastelica et al. (2020) and other imitation learners (Ratliff et al., 2006; Tamar et al., 2016; Lee et al., 2018).

2. Preliminaries

Path planning problem. Let us consider a path planning problem, in particular a point-to-point shortest (*i.e.*, lowest-cost) path problem, on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes representing the locations of the environment and $\mathcal{E}$ is the set of potentially valid movements between nodes. For convenience, we define an alternate graph form $\mathcal{G} = (\mathcal{V}, \mathcal{N})$

Algorithm 1 A* Search
Input: Graph $\mathcal{G}$, movement cost c, start v_{s}, and goal v_{g}
Output: Shortest path P
1: Initialize $\mathcal{O} \leftarrow v_{\mathrm{s}}, \mathcal{C} \leftarrow \emptyset, \texttt{Parent}(v_{\mathrm{s}}) \leftarrow \emptyset$.
2: **while** $v_{\mathrm{g}} \notin \mathcal{C}$ **do**
3:      Select $v^* \in \mathcal{O}$ based on Eq. (1).
4:      Update $O \leftarrow O \setminus v^*, \mathcal{C} \leftarrow \mathcal{C} \cup v^*$.
5:      Extract $\mathcal{V}_{\mathrm{nbr}} \subset \mathcal{V}$ based on Eq. (2).
6:      **for each** $v' \in \mathcal{V}_{\mathrm{nbr}}$ **do**
7:          Update $\mathcal{O} \leftarrow \mathcal{O} \cup v', \texttt{Parent}(v') \leftarrow v^*$.
8:      **end for**
9: **end while**
10: $P \leftarrow \texttt{Backtrack}(\texttt{Parent}, v_{\mathrm{g}})$.

with $\mathcal{N}(v) = \{v' \mid (v, v') \in \mathcal{E}, v \neq v'\}$ referring to the set of the neighbor nodes of v. Each edge (v, v') is given a non-negative movement cost $c(v') \in \mathbb{R}_+$ that depends only on the node v'. Given start and goal nodes, $v_{\mathrm{s}}, v_{\mathrm{g}} \in \mathcal{V}$, the objective of path planning is to find a sequence of connected nodes, $P = (v_1, v_2, \ldots, v_T) \in \mathcal{V}^T$, $v_1 = v_{\mathrm{s}}$, $v_T = v_{\mathrm{g}}$, with its total cost $\sum_{t=1}^{T-1} c(v_{t+1})$ being the lowest among all possible paths from v_{s} to v_{g}. Following Choudhury et al. (2018), this work focuses on a popular setting where $\mathcal{G}$ refers to the eight-neighbor grid world and $c(v')$ is a unit cost taking $c(v') = 1$ when v' is passable and $c(v') = \infty$ otherwise, *e.g.*, when v' is occupied by an obstacle.

A* search. Algorithm 1 overviews the implementation of A* search employed in this work. It explores nodes to find a shortest path P by iterating between (1) selecting the most promising node from the list of candidates for constructing a shortest path and (2) expanding neighbors of the selected node to update the candidate list, until the goal v_{g} is selected. More specifically, the node selection (Line 3 of Algorithm 1) is done based on the following criterion:

$$v^* = \arg\min_{v \in \mathcal{O}} (g(v) + h(v)), \qquad (1)$$

where $\mathcal{O} \subset \mathcal{V}$ is an *open list* that manages candidate nodes for the node selection. $g(v)$ refers to the actual total cost accumulating $c(v')$ for the nodes v' along the current best path from v_{s} to v, which is updated incrementally during the search. On the other hand, $h(v)$ is a heuristic function estimating the total cost from v to v_{g}, for which the straight-line distance between v and v_{g} is often used in the grid world. All the selected nodes are stored in another list of search histories called *closed list*, $\mathcal{C} \subseteq \mathcal{V}$, as done in Line 4.

In Line 5 of Algorithm 1, we expand the neighboring nodes of v^* as $\mathcal{V}_{\mathrm{nbr}} \subset \mathcal{V}$ based on the following criterion:

$$\mathcal{V}_{\mathrm{nbr}} = \{v' \mid v' \in \mathcal{N}(v^*) \wedge v' \notin \mathcal{O} \wedge v' \notin \mathcal{C}\}. \qquad (2)$$

The neighbor nodes $\mathcal{V}_{\mathrm{nbr}}$ are then added to $\mathcal{O}$ in Line 7 to propose new selection candidates in the next iteration. The search is terminated once v_{g} is selected in Line 3 and stored in $\mathcal{C}$, followed by $\texttt{Backtrack}$ that traces the parent nodes $\texttt{Parent}(v)$ from v_{g} to v_{s} to obtain a solution path P.

Data-driven planning setup. As explained in Sec. 1, we seek a principled search-based approach to two distinct scenarios of data-driven path planning in existing work: (1) finding near-optimal paths efficiently for point-to-point shortest path problems (Choudhury et al., 2018), and (2) enabling the path planning on raw images in the absence of movement costs (Vlastelica et al., 2020).

To this end, we abstract these two settings by introducing a 2D environmental-map variable X representing the input graph $\mathcal{G}$. Specifically, corresponding to the two settings with known and unknown movement costs $c(v')$, the map X represents $c(v')$ either (1) explicitly as a binary map $X \in \{0, 1\}^{\mathcal{V}}$ taking 1 for passable locations (*i.e.*, $c(v') = 1$) and 0 otherwise, or (2) implicitly as a raw colored image $X \in [0, 1]^{3 \times \mathcal{V}}$. As a result, each problem instance is represented consistently as an indexed tuple $Q^{(i)} = (X^{(i)}, v_{\mathrm{s}}^{(i)}, v_{\mathrm{g}}^{(i)})$.

For each problem instance $Q^{(i)}$, we further assume that a ground-truth path is given as a 2D binary map $\bar{P}^{(i)} \in \{0, 1\}^{\mathcal{V}}$ whose elements take 1 along the desired path. When $X^{(i)}$ is a binary map indicating the movement cost explicitly, $\bar{P}^{(i)}$ is obtained by solving the shortest path problem using an optimal planner. If $X^{(i)}$ is a raw image, we assume that $\bar{P}^{(i)}$ is provided by a human annotator.

3. Neural A* Search

Now we present the proposed Neural A* search. At a high level, Neural A* uses an encoder to transform a problem instance $Q^{(i)}$ into a *guidance map*, as illustrated in Fig. 2. This guidance map imposes a *guidance cost* $\phi^{(i)}(v) \in \mathbb{R}_+$ to each node v. Then, the differentiable A* module, detailed in Sec. 3.1, performs a search under the policy to minimize a total guidance cost, *i.e.*, $\sum_{t=1}^{T-1} \phi^{(i)}(v_{t+1})$. By repeating this forward pass and the back-propagation through the training procedure described in Sec. 3.2, the encoder learns to capture visual cues in the training instances to improve the path accuracy and search efficiency.

3.1. Differentiable A* Module

Variable representations. To reformulate A* search in a differentiable fashion, we represent the variables in Algorithm 1 as matrices of the map size so that each line can be executed via matrix operations. Specifically, let $O, C, V_{\mathrm{nbr}} \in \{0, 1\}^{\mathcal{V}}$ be binary matrices indicating the nodes contained in $\mathcal{O}, \mathcal{C}, \mathcal{V}_{\mathrm{nbr}}$, respectively (we omit an index i here for simplicity.) We represent the start node

v_s, goal node v_g, and selected node v^* as one-hot indicator matrices $V_s, V_g, V^* \in \{0,1\}^{\mathcal{V}}$, respectively, where $\langle V_s, \mathbb{1} \rangle = \langle V_g, \mathbb{1} \rangle = \langle V^*, \mathbb{1} \rangle = 1$ (*i.e.*, one-hot), $\langle A, B \rangle$ is the inner product of two matrices A and B, and $\mathbb{1}$ is the all-ones matrix of the map size. In addition, let $G, H, \Phi \in \mathbb{R}^{\mathcal{V}}_+$ be a matrix version of $g(v)$, $h(v)$, and $\phi(v)$, respectively.

Node selection. Performing Eq. (1) in a differentiable manner is non-trivial as it involves a discrete operation. Here, we leverage a discretized activation inspired by Hubara et al. (2016) and reformulate the equation as follows:

$$V^* = \mathcal{I}_{\max}\left(\frac{\exp(-(G+H)/\tau) \odot O}{\langle \exp(-(G+H)/\tau), O \rangle} \right), \qquad (3)$$

where $A \odot B$ denotes the element-wise product, and τ is a temperature parameter that will be defined empirically. $\mathcal{I}_{\max}(A)$ is the function that gives the argmax index of A as a binary one-hot matrix during a forward pass, while acting as the identity function for back-propagation. The open-list matrix O is used here to mask the exponential term so that the selection is done from the nodes in the current open list.

Node expansion. Expanding the neighboring nodes of v^* in Eq. (2) involves multiple conditions, *i.e.*, $v' \in \mathcal{N}(v^*)$, $v' \notin \mathcal{O}$, and $v' \notin \mathcal{C}$. Inspired by Tamar et al. (2016), we implement $\mathcal{N}$ as a convolution between V^* and the fixed kernel $K = [[1,1,1]^\top, [1,0,1]^\top, [1,1,1]^\top]$. When X is given as a binary cost map indicating the passable/impassable nodes, V_{nbr} is obtained by the following matrix operations:

$$V_{nbr} = (V^* * K) \odot X \odot (\mathbb{1} - O) \odot (\mathbb{1} - C), \quad (4)$$

where $A * B$ is the 2D convolution of A and B. The multiplication by $X \odot (\mathbb{1} - O) \odot (\mathbb{1} - C)$ acts as a mask to extract the nodes that are passable and not contained in the open nor close lists. Masking with X preserves the original obstacle structures and thus the graph topology, keeping the differentiable A* *complete* (*i.e.*, guaranteed to always find a solution if one exists in the graph $\mathcal{G}$, as in standard A*). We also introduce $\bar{V}_{nbr} = (V^* * K) \odot X \odot O \odot (\mathbb{1} - C)$ indicating the neighboring nodes already in the open list, which we will use to update G below. When X is otherwise a raw image that does not explicitly indicate the passable nodes, we use $V_{nbr} = (V^* * K) \odot (\mathbb{1} - O) \odot (\mathbb{1} - C)$ and $\bar{V}_{nbr} = (V^* * K) \odot O \odot (\mathbb{1} - C)$.

Updating G. As explained earlier, $g(v)$ here represents the total guidance cost paid for the actual path from v_s to v, instead of accumulating the original movement cost $c(v')$ that is not always available explicitly. To update this total cost at each iteration, we partially update G with new costs

G' for the neighbor nodes as follows:

$$
\begin{aligned}
G \quad &\leftarrow \quad G' \odot V_{nbr} + \min(G', G) \odot \bar{V}_{nbr} \\
&\quad + G \odot (\mathbb{1} - V_{nbr} - \bar{V}_{nbr}), \qquad (5) \\
G' \quad &= \quad \langle G, V^* \rangle \cdot \mathbb{1} + \Phi. \qquad (6)
\end{aligned}
$$

In Eq. (5), the neighbors V_{nbr} that are opened for the first time are assigned G', and the neighbors $\bar{V}_{nbr}$ that have already been opened are assigned the lower of the new costs G' and previously computed costs G. The new costs G' for the neighbors are computed in Eq. (6) as the sum of the current cost of the selected node, $g(v^*)$, expressed using $g(v^*) = \langle G, V^* \rangle$, and the one-step guidance cost to the neighbors represented by Φ.

3.2. Training Neural A*

Loss design. The differentiable A* module connects its guidance-map input Φ and search output so that a loss for the output is back-propagated to Φ and, consequently, to the encoder. Here, the output is the closed list C, which is a binary matrix accumulating all the searched nodes V^* from Eq. (3) in a search (see Eq. (8) for details). We evaluate the mean L_1 loss between C and the ground-truth path map $\bar{P}$:

$$\mathcal{L} = \|C - \bar{P}\|_1 / |\mathcal{V}|. \qquad (7)$$

This loss supervises the node selections by penalizing both (1) the false-negative selections of nodes that should have been taken into C to find $\bar{P}$ and (2) the false-positive selections of nodes that were excessive in C to match with $\bar{P}$. In other words, this loss encourages Neural A* to (1) search for a path that is close to the ground-truth path (2) with fewer node explorations.

In practice, we disable the gradients of $\mathcal{O}$ in Eq. (3), $V_{nbr}, \bar{V}_{nbr}$ in Eq. (5), and G in Eq. (6) by detaching them from back-propagation chains. Doing so effectively informs the encoder of the importance of the guidance cost Φ in Eq. (6) for the node selection in Eq. (3), while simplifying recurrent back-propagation chains to stabilize the training process and reduce large memory consumption[2].

Encoder design. The loss shown above is back-propagated through every search step in the differentiable A* module to the encoder. Here, we expect the encoder to learn visual cues in the given problem instances for enabling accurate and efficient search-based planning. These cues include, for instance, the shapes of dead ends and by-passes in binary cost maps or colors and textural patterns of passable roads in raw natural images. For this purpose, we utilize a fully-convolutional network architecture such

---

[2]We found in our experiments that fully enabling the gradients caused training failures due to an out-of-memory problem with 16GB GPU RAM.

as U-Net (Ronneberger et al., 2015) used for semantic segmentation, which can learn local visual representations at the original resolution useful for the downstream task[3]. The input to the encoder is given as the concatenation of X and $V_s + V_g$. In this way, the extraction of those visual cues is properly conditioned by the start and goal positions.

Enabling mini-batch training. The complete algorithm of Neural A* is summarized in Algorithm 2. To accelerate the training, it is critical to process multiple problem instances at once in a mini batch. However, this is not straightforward because those intra-batch samples may be solved within different numbers of search steps. We address this issue by introducing a binary goal verification flag $\eta^{(i)} = 1 - \langle V_g^{(i)}, V^{*(i)} \rangle$ and updating $O^{(i)}, C^{(i)}$ as follows:

$$O^{(i)} \leftarrow O^{(i)} - \eta^{(i)} V^{*(i)}, \quad C^{(i)} \leftarrow C^{(i)} + \eta^{(i)} V^{*(i)}. \quad (8)$$

Intuitively, these operations keep $O^{(i)}$ and $C^{(i)}$ unchanged once the goal is found. We repeat Lines 9–15 until we obtain $\eta^{(i)} = 0$ for all the samples in the batch.

4. Experiments

In this section, we first conduct an extensive experiment to evaluate the effect of Neural A* on the search optimality and efficiency trade-off, *i.e.*, how efficiently Neural A* search can find near-optimal paths for point-to-point shortest path problems. Due to space limitations, we provide the detailed experimental setups in Sec. A of the supplementary material.

4.1. Datasets

We adopted the following public path-planning datasets with obstacle annotations to collect planning demonstrations.

- **Motion Planning (MP) Dataset**: A collection of eight types of grid-world environments with distinctive obstacle shapes, created by Bhardwaj et al. (2017). Each environment group consists of 800 training, 100 validation, and 100 test maps, with the same type of obstacles placed in different layouts. We resized each environmental map into the size of 32×32 to complete the whole experiment in a reasonable time. By following the original setting, the training and evaluation were conducted for each environment group independently.

- **Tiled MP Dataset**: Our extension to the MP dataset to make obstacle structures more complex and diverse. Specifically, we tiled four maps drawn randomly from

---

[3]As the nature of convolutional neural networks, guidance cost outputs are only sensitive to the input map within their limited receptive fields. Thus, when the map size is intractably large, we could partially sweep the input map with the encoder incrementally during a search without changing search results.

Algorithm 2 Neural A* Search

Input: Problem instances $\{Q^{(i)} = (X^{(i)}, v_s^{(i)}, v_g^{(i)}) \mid i = 1, \dots, b\}$ in a mini-batch of size b.
Output: Closed-list matrices $\{C^{(i)} \mid i = 1, \dots, b\}$ and solution paths $\{P^{(i)} \mid i = 1, \dots, b\}$.
  1: **for all** $i = 1, \dots, b$ **do in parallel**
  2:     Compute $V_s^{(i)}, V_g^{(i)}$ from $v_s^{(i)}, v_g^{(i)}$.
  3:     Compute $\Phi^{(i)}$ from $X^{(i)}, V_s^{(i)}, V_g^{(i)}$ by the encoder.
  4:     Initialize $O^{(i)} \leftarrow V_s^{(i)}, C^{(i)} \leftarrow \mathbf{0}, G^{(i)} \leftarrow \mathbf{0}$.
  5:     Initialize $\mathtt{Parent}^{(i)}(v_s^{(i)}) \leftarrow \emptyset$.
  6: **end for**
  7: **repeat**
  8:     **for all** $i = 1, \dots, b$ **do in parallel**
  9:         Select $V^{*(i)}$ based on Eq. (3).
 10:         Compute $\eta^{(i)} = 1 - \langle V_g^{(i)}, V^{*(i)} \rangle$.
 11:         Update $O^{(i)}$ and $C^{(i)}$ based on Eq. (8).
 12:         Compute $V_{\mathrm{nbr}}^{(i)}$ based on Eq. (4).
 13:         Update $O^{(i)} \leftarrow O^{(i)} + V_{\mathrm{nbr}}^{(i)}$.
 14:         Update $G^{(i)}$ based on Eq. (5) and Eq. (6).
 15:         Update $\mathtt{Parent}^{(i)}$ based on Algorithm 1-L6,7.
 16:     **end for**
 17: **until** $\eta^{(i)} = 0$ for $i = 1, \dots, b$
 18: **for all** $i = 1, \dots, b$ **do in parallel**
 19:     $P^{(i)} \leftarrow \mathtt{Backtrack}(\mathtt{Parent}^{(i)}, v_g^{(i)})$.
 20: **end for**

---

the MP dataset to construct an environmental map with the size of 64×64. We repeated this process to create 3,200 training, 400 validation, and 400 test maps.

- **City/Street Map (CSM) Dataset**: A collection of 30 city maps with explicit obstacle annotations as binary images, organized by Sturtevant (2012). For data augmentation, we cropped multiple random patches with the size of 128×128 from each map and resized them to the size of 64×64. We used 20 of the 30 maps to generate random 3,200 training and 400 validation maps and the remaining 10 maps for 400 test samples. In this way, we ensured that no maps were shared between training/validation and test splits.

For each map, we created problem instances by randomly picking out a single goal from one of the four corner regions of the map as well as one, six, and fifteen start locations for training, validation, and test splits, respectively, from areas sufficiently distant from the goal. We obtained the ground-truth shortest paths using the Dijkstra algorithm.

4.2. Methods and Implementations

Neural A*. As the encoder, we adopted U-Net (Ronneberger et al., 2015) with the VGG-16 backbone (Simonyan & Zisserman, 2015) implemented by Yakubovskiy (2020), where the final layer was activated by the sigmoid function to constrain guidance maps to be $\Phi \in [0, 1]^{\mathcal{V}}$. For the differentiable A* module, we used the Chebyshev distance as the heuristic function H suitable for the eight-neighbor grid-world (Patel, 1997). The Euclidean distance multiplied by a small constant (0.001) was further added to H for tie-breaking. The temperature τ in Eq. (3) was empirically set to the square root of the map width.

Baselines. For assessing the significance of Neural A* in search-based planning, we adopted the following data-driven search-based planners as baseline competitors that have the planning success guarantee, like ours, by design. We extended the authors' implementations available online.

- **SAIL** (Choudhury et al., 2018): A data-driven best-first search method that achieved high search efficiency by learning a heuristic function from demonstrations. Unlike Neural A*, SAIL employs hand-engineered features such as the straight-line distances from each node to the goal and the closest obstacles. We evaluated two variants of SAIL where training samples were rolled out using the learned heuristic function (**SAIL**) or the oracle planner (**SAIL-SL**).

- **Blackbox Differentiation** (Vlastelica et al., 2020): A general-purpose approach to data-driven combinatorial solvers including search-based path planning. It consists of an encoder module that transforms environmental maps into node-cost maps and a solver module that performs a combinatorial algorithm as a piecewise constant black-box function taking the cost maps as input. We implemented a Blackbox-Differentiation version of A* search (**BB-A***) by treating our differentiable A* module as a black-box function and training the encoder via black-box optimization, while keeping other setups, such as the encoder architecture and loss function, the same as those of Neural A*.

We also evaluated best-first search (**BF**) and weighted A* search (**WA***) (Pohl, 1970) as baseline classical planners, which used the same heuristic function as that of Neural A*. Finally, we implemented a degraded version of Neural A* named **Neural BF**, which always used $G = \Phi$ in Eq. (3) instead of G incrementally updated by Eq. (5). By doing so, Neural BF ignored the accumulation of guidance costs from the start node, much like best-first search.

4.3. Experimental Setups

All the models were trained using the RMSProp optimizer, with the mini-batch size, the number of epochs, and the learning rate set to (100, 100, 0.001) for the MP dataset and (100, 400, 0.001) for the Tiled MP and CSM datasets.

For each trained model, we calculated the following metrics to evaluate how much the trade-off between search optimality and efficiency was improved from a standard A* search performed using the identical heuristic function.

- **Path optimality ratio (Opt)** measuring the percentage of shortest path predictions for each environmental map, as used by Vlastelica et al. (2020).

- **Reduction ratio of node explorations (Exp)** measuring the number of search steps reduced by a model compared to the standard A* search in $0 - 100$ (%). Specifically, by letting E^* and E be the number of node explorations by A* and a model, respectively, it is defined by $\max(100 \times \frac{E^* - E}{E^*}, 0)$ and averaged over all the problem instances for each environmental map.

- **The Harmonic mean (Hmean)** of Opt and Exp, showing how much their trade-off was improved by a model.

During the training, we saved model weights that marked the best Hmean score on the validation split and used them for measuring final performances on the test split. To investigate statistical differences among models, we computed the bootstrap mean and 95% confidence bounds per metric.

4.4. Results

Comparisons with baselines. Table 1 summarizes quantitative results. Overall, Neural A* outperformed baseline methods in terms of both Opt and Hmean and improved the path optimality and search efficiency trade-off. SAIL and SAIL-SL sometimes performed more efficiently, which however came with low optimality ratios especially for larger and more complex maps of the Tiled MP and CSM datasets. BB-A* showed higher Hmean scores than those of SAIL/SAIL-SL but was consistently outperformed by Neural A*. Since the only difference between Neural A* and BB-A* is whether making the A* search differentiable or treating it as a black-box in updating an identically-configured encoder, the comparison between them highlights the effectiveness of our approach. With the differentiable A* module, Neural A* has access to richer information of internal steps, which is necessary to effectively analyze causal relationships between individual node selections by Eq. (3) and results. This information is however black-boxed in BB-A*. Also, we found that classical heuristic planners (BF and WA*) performed comparably to or
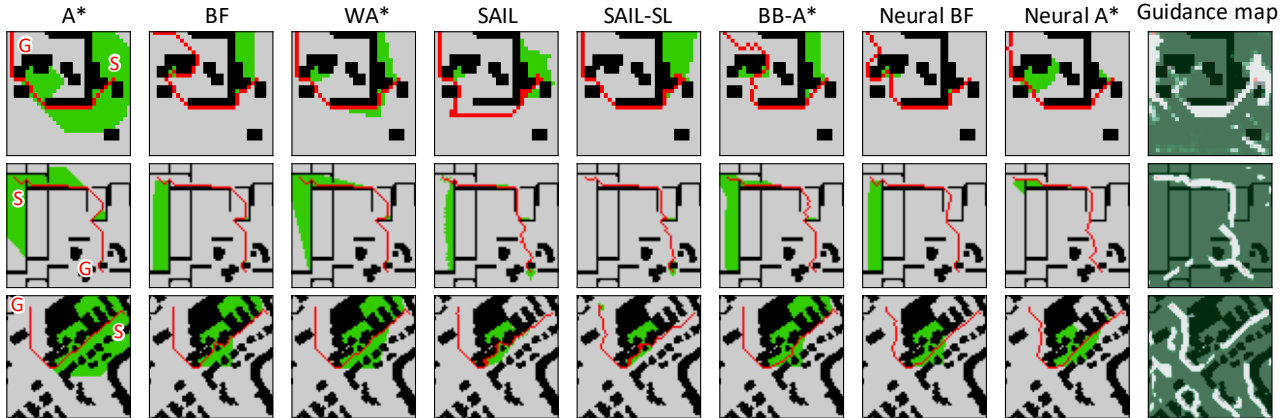
Figure 3. **Selected Path Planning Results.** Black pixels indicate obstacles. Start nodes (indicated by "S"), goal nodes (indicated by "G"), and found paths are annotated in red. Other explored nodes are colored in green. In the rightmost column, guidance maps are overlaid on the input maps where regions with lower costs are visualized in white. More results are presented in Sec. B of the supplementary material.

sometimes better than other data-driven baselines. This result could be explained by our challenging experimental setup adopting randomized start and goal locations instead of pre-defined ones by the original work (Choudhury et al., 2018; Vlastelica et al., 2020). Finally, Neural BF achieved the highest Exp scores in the Tiled MP and CSM datasets but often ended up with sub-optimal paths since it ignored the actual cost accumulation. For more evaluations including analysis on the complexity and runtime of Neural A*, see Sec. B of the supplementary material.

Qualitative results. Figure 3 visualizes example search results with guidance maps produced by the encoder of Neural A*. We confirmed that the encoder successfully captured visual cues in problem instances. Specifically, higher guidance costs (shown in green) were assigned to the whole obstacle regions creating dead ends, while lower costs (shown in white) were given to bypasses and shortcuts that guided the search to the goal. Figure 4 depicts how Neural A* performed a search adaptively for different start and goal locations in the same map. Notice the entrance to the U-shaped dead end is (a) blocked with high guidance costs when the dead end is placed between the start and goal and (b) open when the goal is inside the dead end.

Ablation study. We further assessed the effects of the encoder by comparing several different configurations for its architecture. In the configuration **w/o start and goal**, we fed only an environmental map X_i to the encoder as input. In **w/ ResNet-18 backbone**, a residual network architecture (He et al., 2016) was used instead of VGG-16. Table 2 shows the degraded performances with these configurations especially in terms of the optimality ratio. Additionally, we tried the following variants of the proposed approach, which we found not effective. Adopting the straight-through Gumbel-
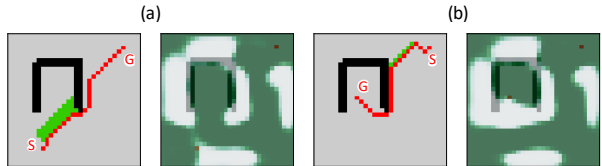

Figure 4. **Adaptive Encoding Results.** (a) The U-shaped dead-end obstacle is placed between the start and goal nodes; (b) The goal node is located inside the U-shaped dead end.

softmax (Jang et al., 2017) for Eq. (3) caused complete planning failures (with Opt, Exp, Hmean all 0) as it biases the planner to avoid the most promising nodes. Using the mean squared loss for Eq. (7) produces the same gradients as the mean L_1 loss up to a constant factor for binary variables (*i.e.*, C and $\bar{P}$), and indeed led to a comparable performance.

Limitations. Although Neural A* worked well in various environments, its current implementation assumes the grid world environments with unit node costs. One interesting direction is to extend Neural A* to work on a high-dimensional or continuous state space. This will require the encoder to be tailored to such a space, as done in Qureshi et al. (2019); Chen et al. (2020); Ichter & Pavone (2019). We leave this extension for future work.

5. Path Planning on Raw-Image Inputs

As another scenario for Neural A*, we address the task of planning paths directly on raw image inputs. Specifically, suppose a video of an outdoor scene taken by a stationary surveillance camera. Planning demonstrations then consist of color images of the scene and actual trajectories of pedestrians (*i.e.*, ground-truth paths provided by human an-

Table 1. **Quantitative Results.** Bootstrap means and 95% confidence bounds of path optimality ratio (Opt), reduction ratio of node explorations (Exp), and their harmonic mean (Hmean).

MP DATASET			
	Opt	Exp	Hmean
BF	65.8 (63.8, 68.0)	44.1 (42.8, 45.5)	44.8 (43.4, 46.3)
WA *	68.4 (66.5, 70.4)	35.8 (34.5, 37.1)	40.4 (39.0, 41.8)
SAIL	34.6 (32.1, 37.0)	**48.6 (47.2, 50.2)**	26.3 (24.6, 28.0)
SAIL-SL	37.2 (34.8, 39.5)	46.3 (44.8, 47.8)	28.3 (26.6, 29.9)
BB-A*	62.7 (60.6, 64.9)	42.0 (40.6, 43.4)	42.1 (40.5, 43.6)
Neural BF	75.5 (73.8, 77.1)	45.9 (44.6, 47.2)	**52.0 (50.7, 53.4)**
Neural A*	**87.7 (86.6, 88.9)**	40.1 (38.9, 41.3)	**52.0 (50.7, 53.3)**
TILED MP DATASET			
	Opt	Exp	Hmean
BF	32.3 (30.0, 34.6)	58.9 (57.1, 60.8)	34.0 (32.1, 36.0)
WA*	35.3 (32.9, 37.7)	52.6 (50.8, 54.5)	34.3 (32.5, 36.1)
SAIL	5.3 (4.3, 6.1)	58.4 (56.6, 60.3)	7.5 (6.3, 8.6)
SAIL-SL	6.6 (5.6, 7.6)	54.6 (52.7, 56.5)	9.1 (7.9, 10.3)
BB-A*	31.2 (28.8, 33.5)	52.0 (50.2, 53.9)	31.1 (29.2, 33.0)
Neural BF	43.7 (41.4, 46.1)	**61.5 (59.7, 63.3)**	44.4 (42.5, 46.2)
Neural A*	**63.0 (60.7, 65.2)**	55.8 (54.1, 57.5)	**54.2 (52.6, 55.8)**
CSM DATASET			
	Opt	Exp	Hmean
BF	54.4 (51.8, 57.0)	39.9 (37.6, 42.2)	35.7 (33.9, 37.6)
WA*	55.7 (53.1, 58.3)	37.1 (34.8, 39.3)	34.4 (32.6, 36.3)
SAIL	20.6 (18.6, 22.6)	41.0 (38.8, 43.3)	18.3 (16.7, 19.9)
SAIL-SL	21.4 (19.4, 23.3)	39.3 (37.1, 41.6)	17.6 (16.1, 19.1)
BB-A*	54.4 (51.8, 57.1)	40.0 (37.7, 42.3)	35.6 (33.8, 37.4)
Neural BF	60.9 (58.5, 63.3)	**42.1 (39.8, 44.3)**	40.6 (38.7, 42.6)
Neural A*	**73.5 (71.5, 75.5)**	37.6 (35.5, 39.7)	**43.6 (41.7, 45.5)**

notators). Given these data, we aim by Neural A* to predict realistic trajectories consistent with those of pedestrians when start and goal locations are provided. We here compared Neural A* with BB-A* (Vlastelica et al., 2020) as a competitor that can also perform planning on raw image inputs. For more comparisons with other imitation learners (Ratliff et al., 2006; Tamar et al., 2016; Lee et al., 2018), see Sec. B.1 of the supplementary material.

Dataset. We used Stanford Drone Dataset (SDD) (Robicquet et al., 2016), which comprises surveillance videos captured by static drone cameras placed at eight distinct locations. We split the dataset in two ways: (1) intra-scenes: for each location, choosing one video to build a single test split while using the rest to train a model, to see if the model can predict trajectories of unseen individuals observed at different times; (2) inter-scenes: performing leave-one-location-out cross-validation to see how well a learned model can generalize to unseen locations. As planning demonstrations, we extracted pedestrian trajectories and the local patch of a video frame that encompassed the trajectories.

Table 2. **Ablation Study.** Performance comparisons with different architectural design choices for the encoder on the MP dataset.

	Opt	Exp	Hmean
Neural A*	**87.7 (86.6, 88.9)**	40.1 (38.9, 41.3)	**52.0 (50.7, 53.3)**
w/o start and goal	67.0 (65.1, 68.8)	36.8 (35.6, 38.1)	41.5 (40.2, 42.7)
w/ ResNet-18 backbone	79.8 (78.1, 81.5)	**41.4 (40.2, 42.7)**	49.2 (47.9, 50.5)

Table 3. **Quantitative Results on SDD.** Bootstrap means and 95% confidence bounds of the chamfer distance between predicted and ground-truth pedestrian trajectories.

	Intra-scenes	Inter-scenes
BB-A*	152.2 (144.9, 159.1)	134.3 (132.1, 136.4)
Neural A*	**16.1 (13.2, 18.8)**	**37.4 (35.8, 39.0)**

Implementations and experimental setups. Unlike the previous experiment with obstacle locations given explicitly, each model now has to learn visual representations of obstacles to avoid them during node selections. Therefore, we modified the U-Net encoder by multiplying the final sigmoid activation by a trainable positive scalar (initialized to 10.0) so that obstacle regions can be assigned sufficiently high costs. We trained both Neural A* and BB-A* using RMSProp with the batch size, the number of epochs, and the learning rate set to $(64, 20, 0.001)$. Because the main objective of this task is to predict paths close to the ground-truth ones, rather than to improve the search optimality and efficiency trade-off, we calculated the chamfer distance as a metric for evaluating the dissimilarities between those paths.

Results. Table 3 shows that Neural A* significantly outperformed BB-A*. As visualized in the first two examples in Fig. 5, Neural A* often predicted paths along roads, resulting in predictions closer to ground-truth paths compared to those by BB-A*. However, both methods sometimes failed to predict actual pedestrian trajectories when there were multiple possible routes to the destinations, as shown in the example at the bottom. A possible extension to address this issue is to adopt a generative framework (Gupta et al., 2018; Salzmann et al., 2020) that allows multiple paths to be stochastically sampled.

6. Related Work

Approaches to path planning problems can be broadly classified into several types, each with its advantages and limitations. For example, sampling-based planning, such as RRT (LaValle, 1998) and PRM (Kavraki et al., 1996), can explore high-dimensional state spaces and has widely been used for robot motion planning (Kingston et al., 2018). A practical challenge is identifying important regions in the state spaces to effectively sample sparse state-points for efficient planning. To this end, a variety of data-driven methods
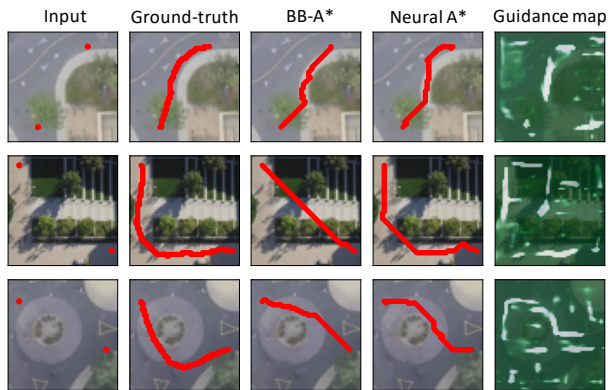
Figure 5. **Path Planning Examples on SDD.** Neural A* predicted paths similar to actual pedestrian trajectories.

have been developed to learn such regions (Ichter et al., 2018; Ichter & Pavone, 2019) or to learn exploration strategies (Qureshi et al., 2019; Pérez-Higueras et al., 2018; Chen et al., 2020) from expert demonstrations or prior successful planning experiences.

Meanwhile, reactive planners learn a policy of moving agents to determine the next best actions, such as moving left or right, from current states via supervised learning (Tamar et al., 2016; Kanezaki et al., 2017; Gupta et al., 2017; Karkus et al., 2017; Lee et al., 2018; Bency et al., 2019) or inverse reinforcement learning (IRL) (Ratliff et al., 2006; Ziebart et al., 2008; Wulfmeier et al., 2015; Kim & Pineau, 2016; Kretzschmar et al., 2016; Fu et al., 2018; Yu et al., 2019). These approaches can be useful for dynamic environments where the agents have to act adaptively. Also, IRL-based methods are relevant to our work in that they learn to recover cost functions from demonstrations. However, they often suffer from planning failures especially for a large map and thus require the help of other classical planners to enable long-range planning (Faust et al., 2018). In Sec. B.1 of the supplemental material, we provide quantitative evaluations of some relevant methods (Ratliff et al., 2006; Tamar et al., 2016; Lee et al., 2018) with our experimental setting.

Compared to these two approaches, search-based planning is advantageous in terms of ensured success in finding valid paths in a fine grid map. Classical heuristic planners have sought better heuristic functions, search algorithms, and efficient implementations to improve the search performance (*e.g.*, Burns et al. (2012); Zhou & Zeng (2015); Abd Algfoor et al. (2015)). Recent data-driven approaches further extend heuristic planners in two ways: learning from expert demonstrations to (a) find near-optimal paths efficiently (Choudhury et al., 2018; Takahashi et al., 2019) or (b) perform the planning directly on raw image inputs (Vlastelica et al., 2020). Standing on both sides, our work proposes the first differentiable search-based planner that can solve these two

problems in a principled fashion. From another perspective, data-driven heuristic planners can learn their cost functions, as in ours and Vlastelica et al. (2020), as well as their heuristic functions, as in Choudhury et al. (2018); Takahashi et al. (2019). These variations are analogous to learning a (negative) reward function in IRL and a Q-function in RL, respectively. Very recently, Archetti et al. (2021) have extended Neural A* to learn both of these functions.

7. Conclusion

We have presented a novel data-driven search-based planner named Neural A*, which involves a differentiable A* algorithm. Neural A* learns from demonstrations to improve the trade-off between search optimality and efficiency in path planning and also to enable the planning directly on raw image inputs. Our extensive experimental evaluations on multiple public datasets have demonstrated the effectiveness of our approach over state-of-the-art planners.

References

Abd Algfoor, Z., Sunar, M. S., and Kolivand, H. A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, 2015.

Archetti, A., Cannici, M., and Matteucci, M. Neural weighted A*: Learning graph costs and heuristics with differentiable anytime A*. *arXiv preprint arXiv:2105.01480*, 2021.

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

Bhardwaj, M., Choudhury, S., and Scherer, S. Learning heuristic search via imitation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2017.

Burns, E. A., Hatem, M., Leighton, M. J., and Ruml, W. Implementing fast heuristic search code. In *Proceedings of the Annual Symposium on Combinatorial Search (SOCS)*, 2012.

Chen, B., Dai, B., Lin, Q., Ye, G., Liu, H., and Song, L. Learning to plan in high dimensions via neural exploration-exploitation trees. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Choudhury, S., Bhardwaj, M., Arora, S., Kapoor, A., Ranade, G., Scherer, S., and Dey, D. Data-driven planning via imitation learning. *International Journal of Robotics Research (IJRR)*, 37(13-14):1632–1672, 2018.

Faust, A., Oslund, K., Ramirez, O., Francis, A., Tapia, L., Fiser, M., and Davidson, J. PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5113–5120, 2018.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

González, D., Pérez, J., Milanés, V., and Nashashibi, F. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2015.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2255–2264, 2018.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2616–2625, 2017.

Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4107–4115, 2016.

Ichter, B. and Pavone, M. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters (RA-L)*, 4(3):2407–2414, 2019.

Ichter, B., Harrison, J., and Pavone, M. Learning sampling distributions for robot motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7087–7094, 2018.

Ichter, B., Schmerling, E., Lee, T. W. E., and Faust, A. Learned critical probabilistic roadmaps for robotic motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9535–9541, 2020.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Kanezaki, A., Nitta, J., and Sasaki, Y. Goselo: Goal-directed obstacle and self-location map for robot navigation using reactive neural networks. *IEEE Robotics and Automation Letters (RA-L)*, 3(2):696–703, 2017.

Karkus, P., Hsu, D., and Lee, W. S. QMDP-Net: Deep learning for planning under partial observability. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4694–4704, 2017.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *t-ro*, 12(4):566–580, 1996.

Kim, B. and Pineau, J. Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics*, 8(1):51–66, 2016.

Kingston, Z., Moll, M., and Kavraki, L. E. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems*, 1: 159–185, 2018.

Kretzschmar, H., Spies, M., Sprunk, C., and Burgard, W. Socially compliant mobile robot navigation via inverse reinforcement learning. *International Journal of Robotics Research (IJRR)*, 35(11):1289–1307, 2016.

LaValle, S. M. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University, 1998.

Lee, L., Parisotto, E., Chaplot, D. S., Xing, E., and Salakhutdinov, R. Gated path planning networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Paden, B., Čáp, M., Yong, S. Z., Yershov, D., and Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

Patel, A. Heuristics – Stanford CS Theory, 1997. URL http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html.

Pérez-Higueras, N., Caballero, F., and Merino, L. Learning human-aware path planning with fully convolutional networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–5, 2018.

Pohl, I. Heuristic search viewed as path finding in a graph. *Artificial intelligence*, 1(3-4):193–204, 1970.

Qureshi, A. H., Simeonov, A., Bency, M. J., and Yip, M. C. Motion planning networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2118–2124, 2019.

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. Maximum margin planning. In *Proceedings of the International Conference on Machine Learning (ICML)*, ICML '06, pp. 729–736, 2006.

Robicquet, A., Sadeghian, A., Alahi, A., and Savarese, S. Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 549–565, 2016.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pp. 234–241, 2015.

Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D. V., and Kragic, D. Dual arm manipulation - a survey. *Robotics and Autonomous Systems*, 60 (10):1340–1353, 2012.

Sturtevant, N. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Game*, 4(2):144 – 148, 2012.

Takahashi, T., Sun, H., Tian, D., and Wang, Y. Learning heuristic functions for mobile robot path planning using deep neural networks. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 764–772, 2019.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2154–2162, 2016.

Vlastelica, M., Paulus, A., Musil, V., Martius, G., and Rolínek, M. Differentiation of blackbox combinatorial solvers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Wulfmeier, M., Ondruska, P., and Posner, I. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Yakubovskiy, P. Segmentation models pytorch. https://github.com/qubvel/segmentation_models.pytorch, 2020.

Yu, L., Yu, T., Finn, C., and Ermon, S. Meta-inverse reinforcement learning with probabilistic context variables. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 11772–11783, 2019.

Zhou, Y. and Zeng, J. Massively parallel a* search on a gpu. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2015.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pp. 1433–1438, 2008.