# Path Planning using Neural A* Search

Lidia Fargueta Pelufo

# Index

- Why to focus on this algorithm?
- Why to introduce learning?
- What’s the difference with other models?
- Two Scenarios
- Schematic Diagram
- Datasets
- VGG-16 Architecture
- Differentiable A* module
- Training
- Summary of the algorithm
- Evaluation for: Point-to-point shortest-path search & Raw images.
- Innovation ideas.

# Why did I choose to focus on this algorithm?

- It is the base to coming diffusion models in path-planning.
- There is an already implemented simulation in a GitHub repository that can help me to learn easier.
- GNNs are a more complicated architectures, RRT* has already been implemented in Duckiebot and RRT* dynamic will require real-time duckiebot and object tracking, more complicated.

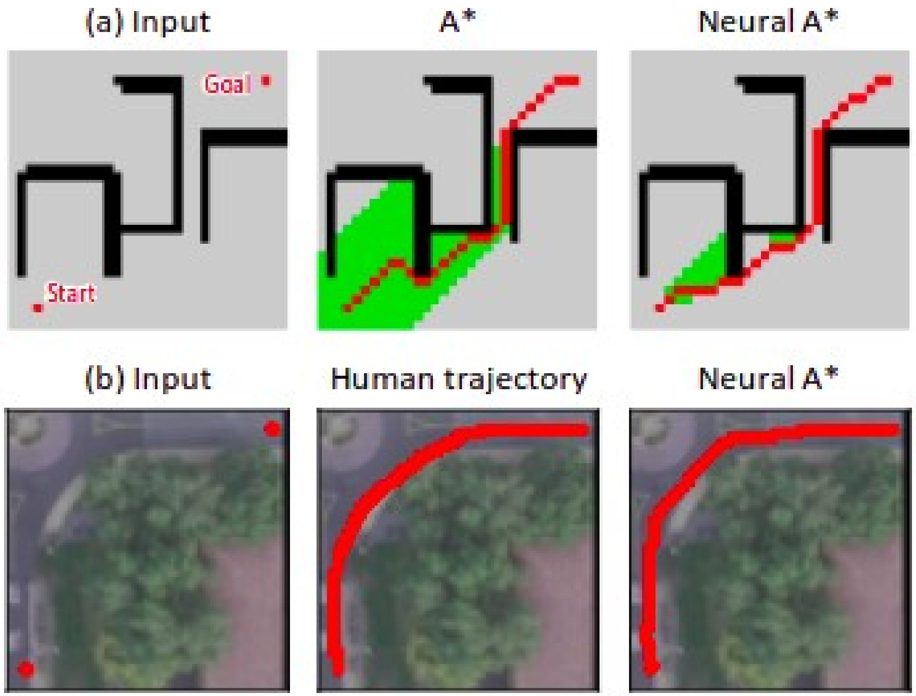
# Why to introduce learning?

- Finding near-optimal paths more efficiently than classical heuristic planners in point-to-point shortest path search problems.
- Enabling path planning on raw image inputs, which is hard for classical planners unless semantic pixel-wise labeling of the environment is given.

# What’s the difference with other models?

- We pursue a search-based approach to data-driven planning with the intrinsic advantage of guaranteed planning success, if one solution exists (compared to sampling-based or reactive planning).
- Studies in this direction so far are largely limited due to the difficulty arising from the discrete nature of incremental search steps in search-based planning, which makes the learning using back-propagation non-trivial. To address the problem, we design a differentiable A* algorithm.

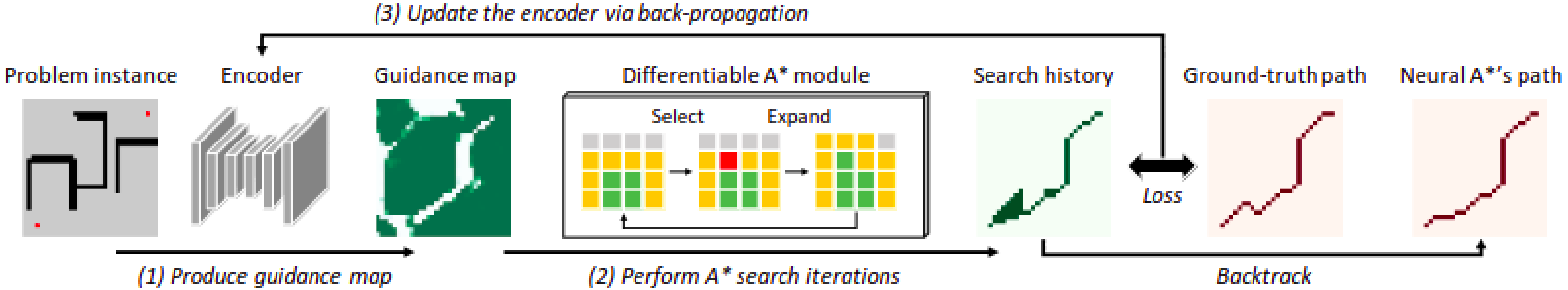
# Two Scenarios




a. Point-to-point shortest path search.

b. Path planning on raw image inputs.

# Schematic Diagram




**Problem instance:**

$$Q^{(i)} = (X^{(i)}, v_{\mathrm{s}}^{(i)}, v_{\mathrm{g}}^{(i)})$$

Where:
- X is the 2D environmental-map variable.
- v_s is the starting point.
- v_g is the goal point.

**Encoder main characteristics:**
- U-Net with the VGG-16 backbone.

**Differentiable A*:**
More details in the following slides.

**Loss:**

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

Where:
- C is the prediction
- P is the ground-truth path
- V contains all the nodes

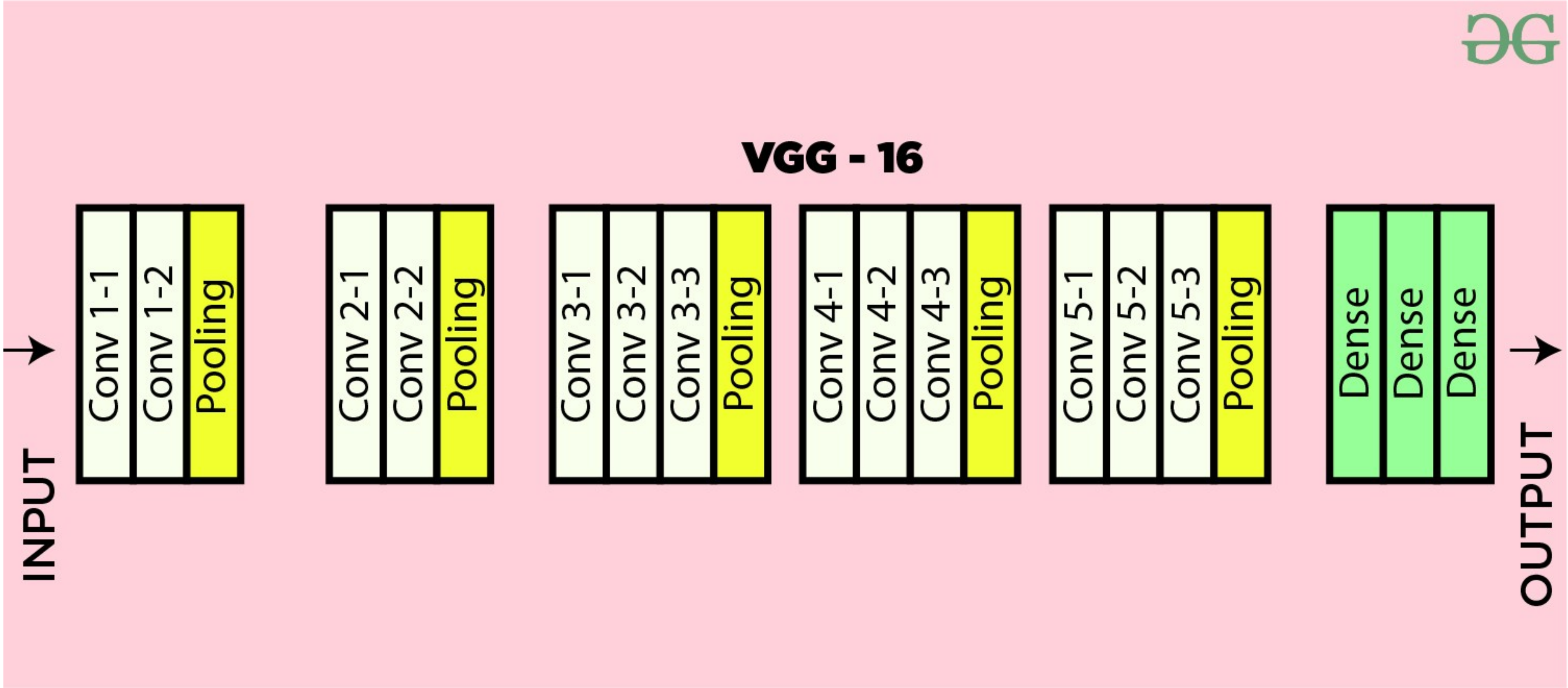
# Datasets

- For Point-to-point path search*
  - Motion Planning Dataset (MP)
  - Tiled MP Dataset
  - City/Street Map (CSM) Dataset

  * Ground-truth shortest paths with Dijkstra algorithm.

- Raw image inputs
  - Stanford Drone Dataset

# VGG-16 Architecture



# Differentiable A* module

- **Variables representations**:
  - $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where G is the graph, V is the set of nodes representing the locations in the environment and E is the set of potentially valid movements between nodes.
  - $\mathcal{N}(v) = \{v' \mid (v, v') \in \mathcal{E}, v \neq v'\}$ , set of neighbor nodes of node v.
  - (v, v') is an edge.
  - $\mathcal{O} \subset \mathcal{V}$ open list with the candidate nodes for the node selection.
  - $\mathcal{C} \subseteq \mathcal{V}$, closed list (selected nodes) --> Output of the A* module.
  - $O, C, V_{\text{nbr}} \in \{0, 1\}^{\mathcal{V}}$, binary matrices indicating the nodes contained in $\mathcal{O}, \mathcal{C}, \mathcal{V}_{\text{nbr}}$

# Differentiable A* module

- **Variables representations**:
  - $V_{\mathrm{s}}, V_{\mathrm{g}}, V^* \in \{0,1\}^{\mathcal{V}}$ matrices to represent the start node, goal node and selected node.
  - In A* algorithm, to select the next node of the path, we follow this criterion:

    $$v^* = \arg\min_{v \in \mathcal{O}} \left( g(v) + h(v) \right)$$

    - g(v) is the actual total cost accumulating c(v') for the nodes v' along the current best path from v_s (start) to v (current node).
    - h(v) is a heuristic function estimating the total cost from v to v_g (goal)
  - $G, H, \Phi \in \mathbb{R}_+^{\mathcal{V}}$, matrix version of g(v), h(v) and $\phi(v)$. This last one is the guidance cost to each node.

# Differentiable A* module

- **Node selection:**

The equation is reformulated as: $V^* = \mathcal{I}_{\max}\left(\frac{\exp(-(G+H)/\tau) \odot O}{\langle \exp(-(G+H)/\tau), O \rangle}\right)$ **Eq. 3**

where $\tau$ is a temperature parameter defined empirically and

$\mathcal{I}_{\max}(A)$ is the function that gives the argmax index of A.

- **Node expansion:** (neighboring nodes of v*)

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

where $K = [[1,1,1]^\top, [1,0,1]^\top, [1,1,1]^\top]$.

Neighboring nodes in the open list: $\bar{V}_{\mathrm{nbr}} = (V^* * K) \odot X \odot O \odot (\mathbb{1} - C)$

when X is a raw image...

$$V_{\mathrm{nbr}} = (V^* * K) \odot (\mathbb{1} - O) \odot (\mathbb{1} - C)$$

$$\bar{V}_{\mathrm{nbr}} = (V^* * K) \odot O \odot (\mathbb{1} - C)$$

# Differentiable A* module

- **Updating G: (total guidance cost, updated at each iteration)**

$$\begin{aligned} G &\leftarrow G' \odot V_{\mathrm{nbr}} + \min(G', G) \odot \bar{V}_{\mathrm{nbr}} \\ &\quad + G \odot (\mathbb{1} - V_{\mathrm{nbr}} - \bar{V}_{\mathrm{nbr}}), \end{aligned} \quad \textbf{Eq. 5}$$

$$G' = \langle G, V^* \rangle \cdot \mathbb{1} + \Phi. \quad \textbf{Eq. 6}$$

Guidance map

# Training

- To accelerate the training, we use mini-batch training: process multiple problem instances at once
- PROBLEM: Those intra-batch samples may be solved within different numbers of search steps. We then introduce a binary goal verification flag $\eta^{(i)} = 1 - \langle V_{\mathrm{g}}^{(i)}, V^{*(i)} \rangle$ and update O and C as follows:

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

**Eq. 8**

# Training

- Point-to-point shortest path search:

| Dataset | Optimizer | Mini-batch size | Epochs | Learning rate |
|---|---|---|---|---|
| MP Dataset | RMSProp | 100 | 100 | 0.001 |
| Tiled MP and CSM Datasets | RMSProp | 100 | 400 | 0.001 |

- Raw images:

| Optimizer | Mini-batch size | Epochs | Learning rate | Extras |
|---|---|---|---|---|
| RMSProp | 64 | 20 | 0.001 | - Multiply the final sigmoid activation by a trainable positive scalar (initialized to 10.0).<br>- Chamfer distance as metric for evaluating dissimilarities between prediction and truth |

# Summary of the algorithm

**Algorithm 2** Neural A* Search

**Input:** Problem instances $\{Q^{(i)} = (X^{(i)}, v_s^{(i)}, v_g^{(i)}) \mid i = 1, \ldots, b\}$ in a mini-batch of size $b$.

**Output:** Closed-list matrices $\{C^{(i)} \mid i = 1, \ldots, b\}$ and solution paths $\{P^{(i)} \mid i = 1, \ldots, b\}$.

**for all** $i = 1, \ldots, b$ **do in parallel**<br>
    Compute $V_s^{(i)}, V_g^{(i)}$ from $v_s^{(i)}, v_g^{(i)}$.<br>
    Compute $\Phi^{(i)}$ from $X^{(i)}, V_s^{(i)}, V_g^{(i)}$ by the encoder.<br>
    Initialize $O^{(i)} \leftarrow V_s^{(i)}$, $C^{(i)} \leftarrow \mathbf{0}$, $G^{(i)} \leftarrow \mathbf{0}$.<br>
    Initialize $\texttt{Parent}^{(i)}(v_s^{(i)}) \leftarrow \emptyset$.<br>
**end for**<br>
**repeat**<br>
    **for all** $i = 1, \ldots, b$ **do in parallel**<br>
        Select $V^{*(i)}$ based on Eq. (3).<br>
        Compute $\eta^{(i)} = 1 - \langle V_g^{(i)}, V^{*(i)} \rangle$.<br>
        Update $O^{(i)}$ and $C^{(i)}$ based on Eq. (8).<br>
        Compute $V_{\mathrm{nbr}}^{(i)}$ based on Eq. (4).<br>
        Update $O^{(i)} \leftarrow O^{(i)} + V_{\mathrm{nbr}}^{(i)}$.<br>
        Update $G^{(i)}$ based on Eq. (5) and Eq. (6).<br>
        Update $\texttt{Parent}^{(i)}$ based on Algorithm 1-L6,7.<br>
    **end for**<br>
**until** $\eta^{(i)} = 0$ for $i = 1, \ldots, b$<br>
**for all** $i = 1, \ldots, b$ **do in parallel**<br>
    $P^{(i)} \leftarrow \texttt{Backtrack}(\texttt{Parent}^{(i)}, v_g^{(i)})$.<br>
**end for**

# Evaluation for point-to-point shortest path search

- Metrics to evaluate how much the trade-off between search optimality and efficiency was improved from a standard A* search:
  - Path optimality ratio (Opt).
  - Reduction ratio of node explorations (Exp).
  - The Harmonic mean (Hmean) of Opt and Exp.

# For the point-to-point shortest path...

- We compare with BF, WA*, SAIL, SAIL-SL, BB-A* and Neural BF:
  - Talking about efficiency (Exp), the other algorithms, in general, are more efficient, but at the end Neural A* it always outperforms when talking about their trade-off.
  - Classical heuristic planners performed comparibly or sometimes better than other data-driven baselines --> challenging experimental setup with randomized start and goal locations instead of pre-defined ones.

# For the point-to-point shortest path...

- Limitations:
  - It works with grid world environments with unit node cost.
  - Possible **future research**: Extend it to work on high-dimensional or continuous state space.
- **Own idea:**
  - It was checked with other backbone (ResNet-18), but it didn't improve. We can try to test others.

# For the raw-images...

- We compare with BB-A* and it outperforms.
- Limitations:
  - Both methods sometimes failed to predict actual pedestrian trajectories when there were multiple possible routes.
  - Possible **future research**: Adopt a generative framework. --> It has been done with diffusion models.

# Other ideas for “innovation” but that I checked and were already implemented

- Same but with RRT* algorithm (sample-based planning) --> Neural Informed RRT* (https://arxiv.org/html/2309.14595v2)
- Use ViT architecture instead of CNN --> ViT-A* (https://arxiv.org/abs/2310.07525)
- What about a mix? --> ViT-RRT*? Does it make sense? (this one hasn’t been implemented, I think)
- Maybe problems with Edge computation due to training? Then, simplify architecture?