Enhancing Path Planning through Learning Based Methods

Prathamesh Ringe (BT19MEC122), Rishabh Verma (BT19MEC092), Sania Subhedar (BT19MEC103), Yagnesh Devada (BT19ECE122)

Submitted to Dr Trushar B Gohil

Visvesvaraya National Institute Of Technology, Nagpur

1 Introduction

Path planning is critical for autonomous navigation and mobile robots in general. In order to construct an obstacle-free trajectory, we need to know the map of the environment in which our robot will operate. The problem consists of two steps, Global and Local Planning. The Global Plan gives the path which the robot needs to follow till the goal pose is reached. The local plan aims to provide the motion of the robot while following the global plan, it tends to achieve this in real-time to survive dynamic environments. Many approaches to tackle the global planning challenge have been proposed. Standard solutions such as A* and RRT/RRT* are widely employed. Though each of the methods has its own drawbacks. The trade-off is primarily between optimality and computation. We suggest a way for reducing this trade-off and attempting to improve global planning performance using a Convolutional Neural Network based architecture. In general, both of the methods become computationally heavy as the size of the map increases. The prime reason behind this is being the search method used by these algorithms.

2 Background

2.1 Path Planning

The problem of path planning deals with finding a set of states of an entity such that the path formed is computational complexity, safety and optimality. There are various global path planning algorithms such as

- Dijkstra's
- A*
- RRT
- RRT*
- APF

Algorithm 1 A* Search Algorithm

```
open ← MinHeap()
closed ← Set()
predecessors ← Dict()
open.push(s,0)
while !open.isEmpty() do
    u, uCost ← open.pop()
    if isGoal(u) then
        return extractPath(u,predecessors)
    for all v ∈ u.successors()
        if v ∈ closed then
            continue
        uvCost ← edgeCost(G, u, v)
        if v ∈ open then
            if uCost + uvCost + h(v) < open[v] then
                open[v] ← uCost + uvCost +h(v)
                costs[v] ← uCost + uvCost
                predecessors[v] ← u
        else
            open.push(v, uCost + uvCost)
            costs[v] ← uCost + uvCost
            predecessors[v] ← u
    closed.add(u)
end while
```

Every algorithm has their advantages and disadvantages in terms of computational complexity, safety and optimality.

2.2 Artificial Intelligence

Artificial Intelligence is a branch of Computer Science inspired by the working of our brain to enable machines to replicate human behavior. AI is achieved by first understanding how the human brain works, as well as how humans learn, decide, and work when attempting to solve a problem, and then applying the findings to the development of intelligent software and systems.

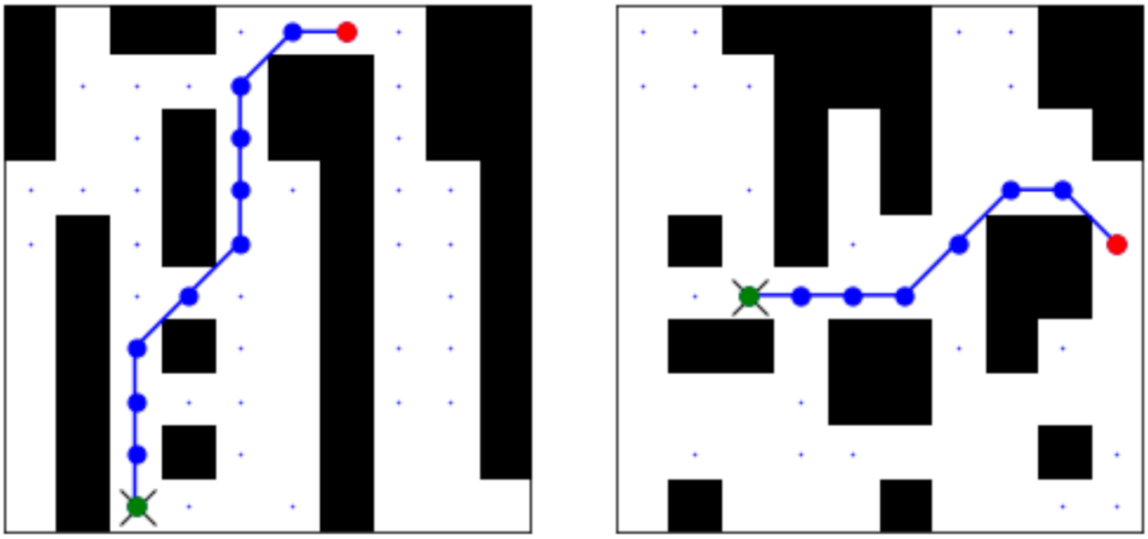

Figure 1: One Shot Path Planning Dataset

- Machine Learning: Machine Learning is a subfield of artificial with a focus on learning through data that is being fed to the machine.
- Deep Learning: This is again a sub field of AI with a focus

2.3 Dataset

We used 30000 samples for each example, each with distinct obstacle layouts and random start and end places. To avoid very short pathways and trivial instances, the minimum Euclidean distance between start and end points was set to 5. For training, 28000 samples were used, and for testing, 2000 samples were used. For the second scenario, we used 1000 15 X 15-size samples with various obstacle layouts but fixed start and end positions as shown in the Fig. 2

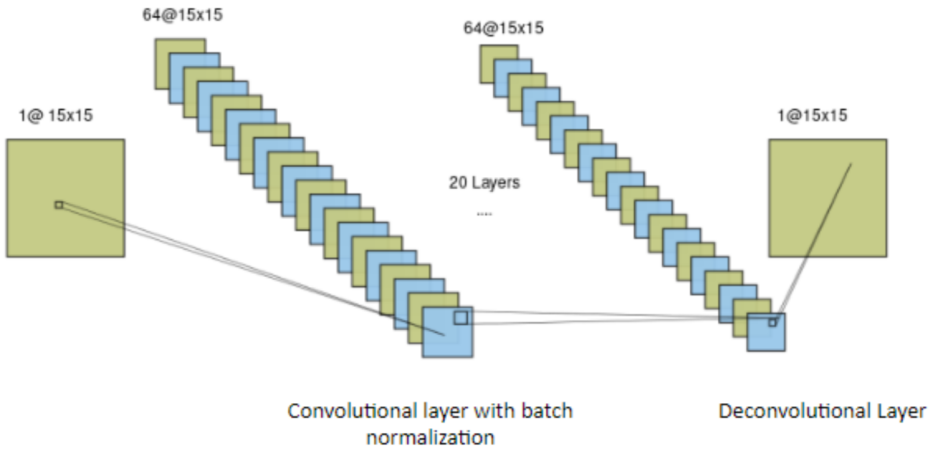

Figure 2: Architecture

2.4 Architecture

We used a fully convolutional network as shown in Fig. 2. The input layer is a 3D input consisting of three 2D binary images of size n × n and the output is a single 2D binary image of size n×n as described above. We used 20 identical convolutional layers with 64 filters of size 3 × 3 and with stride 1. After each convolutional layer we used a batch normalization layer (not shown). After the convolutional layers, we added one deconvolutional layer with one filter of size 3 × 3. This was followed by a batch normalization layer and, in addition, a dropout (10%) layer. In all convolutional layers we used ReLU activation units, whereas in the deconvolutional layer a sigmoid activation unit was used. The ADAM optimiser with default learning parameters and a batch size of 64 samples was used for training the network. Early validation stopping was used if the accuracy of the validation set did not increase within the last 10 epochs to prevent overfitting. We used 28000 samples to train the network (26000 samples for training and 2000 samples for validation) and 2000 samples for testing the network's performance on unseen samples.

2.5 Literature Review

We found some literature addressing the similar issue. Dikjstra's algorithm [1] although provides optimal path proves computationally heavy in large grid-maps.The Neural A* [4] approach uses U-Net which is generally used for semantic labelling. The map is fed to this encoder to generate a guidance map on which the A* search is applied. The extracted path is then back propagated to update the encoder that helps the network to learn how to segment out the map in a way to reduce the search space substantially. Eventually this reduces the computation for the search algorithm. Neural RRT* [3] trains a neural network model is used to initialize the sampling process at first. It outputs a sharp probability distribution of the optimal path for the current map. The probability that quantities of states are selected in the sampling process will approach zero if used directly. Then the probability of completeness of the algorithm is hardly guaranteed. In order to maintain the ability of the NRRT* to maintain the theoretical completeness guarantee, samples are drawn from a uniform sampler. So there are two samplers in the NRRT*, a nonuniform sampler from the neural network model and a uniform sampler. The One-Shot Path Planning [2] approach employs a 20-layer fully convolutional neural network (FCN) with one deconvolutional layer (transpose convolution). In the convolutional layers, there are ReLU activation units and a sigmoid activation unit, and in the deconvolutional layer, there is a sigmoid activation unit (the last layer). Following each convolutional layer, a batch normalisation layer is used, as well as a dropout (10%) layer after the deconvolutional layer.

Hyperparameters
Learning Rate
Batch Size
Filters
Filter Size
Number of Convolutional Layers

Table 1: Table to test captions and labels.

3 Experimentation

Our architecture involved tuning the hyperparameters listed in the Table. 1. These hyperparameters were tuned one by one while keeping the other constant. As for the number of neural network layers we were able to get the best accuracy for 20 CNN layers. The training time exponentially increased as we increased the number of CNN layers from 2 to 20. We obtained maximum accuracy for 20 CNN layers as can be seen in Fig. 3. During the hyperparameter tuning, the model was trained for 100 epochs. At every 10 epochs validation loss was calculated to ensure that the model can learn the features and not memorize data. We used Mean Squared Error (MSE) as a loss function and the optimizer used was ADAM.

The network was implemented using Keras and TensorFlow API. The training process was performed on a computer with AMD Ryzen 5-4600H (3GHz) CPU and NVIDIA GTX 1650Ti (4GB) GPU. The training took 5 hours using the tuned hyperparameters.

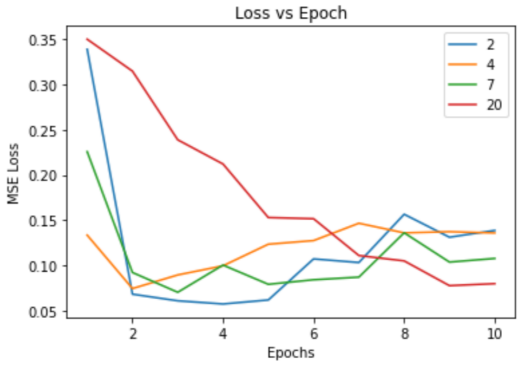

Figure 3: Plot for the loss of 2,4,7 and 20 CNN Layers Architecture

4 Future Works

This is not the case with the run time of FCN. Trajectory planning of a robot consists of two aspects - global and local planning. Our proposed method aims to improve global path planning. The A* search's run time quadratically grows with path length,

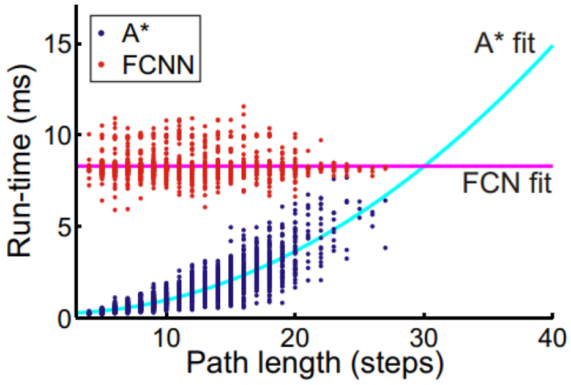

Figure 4: Comparision of Classical A* and FCN A*

Hyperparameters	Value
Learning Rate	0.01
Batch Size	64
Filters	64
Filter Size	3x3
Number of Convolutional Layers	20

Table 2: Table to test captions and labels.

which is directly proportional to the size of the map and the start and target nodes. After achieving substantial improvement for the global planning module, we aim to improve local planning algorithms using the Deep Learning technique - Conv LSTM. These algorithms can be employed to perform autonomous navigation on a ground robot for real-time implementation.

References

[1] Edsger W Dijkstra. "A note on two problems in connexion with graphs". **in***Numerische mathematik*: 1.1 (1959), **pages** 269–271.

[2] Tomas Kulvicius **andothers**. "One-shot path planning for multi-agent systems using fully convolutional neural network". **in***CoRR*: abs/2004.00568 (2020). arXiv: 2004.00568. URL: https://arxiv.org/abs/2004.00568.

[3] Jiankun Wang **andothers**. "Neural RRT*: Learning-Based Optimal Path Planning". **in***IEEE Transactions on Automation Science and Engineering*: 17.4 (2020), **pages** 1748–1758. DOI: 10.1109/TASE.2020.2976560.

[4] Ryo Yonetani **andothers**. "Path Planning using Neural A* Search". **in***Proceedings of the 38th International Conference on Machine Learning*: **byeditor**Marina Meila **and** Tong Zhang. **volume** 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, **pages** 12029–12039. URL: https://proceedings.mlr.press/v139/yonetani21a.html.