From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots

Manzil Roy

Supervisor: Marcell Missura

July 9, 2019

Abstract

This paper is a work on learning from demonstration for motion planning and this report is a critical analysis of it. A **deep residual CNN** is trained to learn a fairly complicated mapping from a feature set to a target data. The feature set is the navigation input which consists of **2D laser range findings** and **target position** while the output is the steering (or navigation) commands for the robot. The important observation in this paper is that training of the CNN is done in a simulation environment and the testing is done in a real scenario. The authors state that this type of supervised learning technique for navigation presented in this paper is one of the first approaches that learns a target oriented end to end navigation model for a robotic platform. A thorough evaluation of the deep CNN model is done and it is compared to the traditional grid based global approach.

1 Paper Summary

1.1 Problem Statement

The problem statement addressed in this paper is navigation of a ground robot, i.e, getting from a current position to a target position. Classical motion planning requires development and testing of algorithms, procurement of a map of the environment, and data pre-processing. That's where the data-driven end to end motion planner comes into picture. A CNN is trained using data and velocity commands (which are the feature and outputs respectively), and is tested as an end to end motion planner.

$$\mathbf{u} = \mathcal{F}_\theta(\mathbf{y}, \mathbf{g}) \tag{1}$$

As per equation (1) , $\mathbf{y}$ is the sensor data obtained from the laser readings and $\mathbf{g}$ is the goal/target to be reached. Together they comprise the function parameters and $\mathbf{u}$ is the steering commands comprising the rotational and translational velocities. A neural network is used as the function approximator, which will learn the network parameter vector. The optimization criterion is based on the difference between the training data (expert steering commands) and the predicted values from the network during training : $|\mathrm{F}_\theta(\mathbf{y}, \mathbf{g}) - \mathbf{u_{exp}}|$.

1.2 Deep Residual Learning

The most crucial part of this paper is the network used to train the robot. It is a Deep Convolutional Neural Network, specifically a ResNet. Before delving into the actual problem statement of training our robot, I would like to focus on the specifics of a ResNet and its importance. CNNs, theoretically can approximate any functions with proper hyper-parameter tuning. It can model complex and highly non-linear dependencies. The network depth is a very significant factor . As per research , a huge improvement in performance is obtained solely on the basis of depth of a network . Some research results are mentioned in [1] to support this fact. A 28 % relative improvement on the COCO detection dataset is obtained solely by the depth. But learning better networks are not as simple as stacking up more layers and constructing deeper networks .

- Increasing the network depth after a certain extent introduces overfitting in our model.
- A degradation has been noticed in deeper networks and overfitting is not the cause for it. Accuracy becomes stagnant.
- Issues like vanishing/exploding gradients creep in which are a hindrance to the convergence of the network.

Addressing the last point above, an identity mapping has been introduced in deep networks, and the other layers are copied from the learned shallower model.

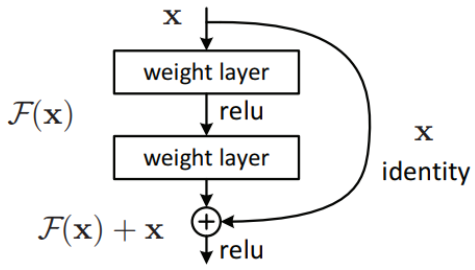


Figure 1: Building Block of a Residual Network

1.3 Structure of our CNN

Our network is a typical CNN with residual connections, as shown in Figure 2. The laser data is processed by the convolutional part which consists of two residual building parts .The Fully Connected part of the network combines the extracted features and the target information. The dimensions of the FC layers are small during training and testing on simulation. Experimentally it has been observed that increasing the dimension of the FC layer during testing in real environment gives better results.

1.4 ROS 2D navigation stack and Training

The ROS 2D navigation stack is used as the expert motion planner. It takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base. The input feature vector for our model is the sensor data and

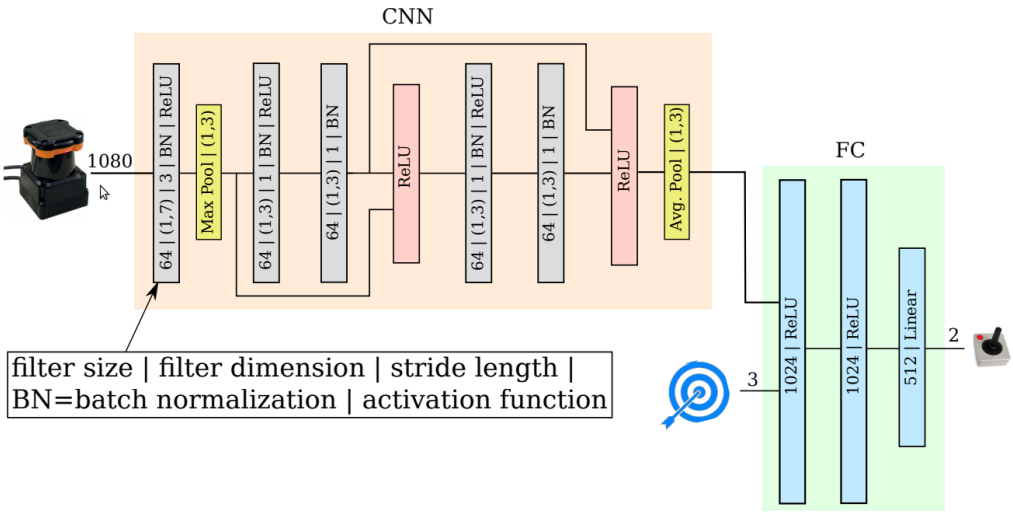


Figure 2: Structure of the Convolutional Neural Network

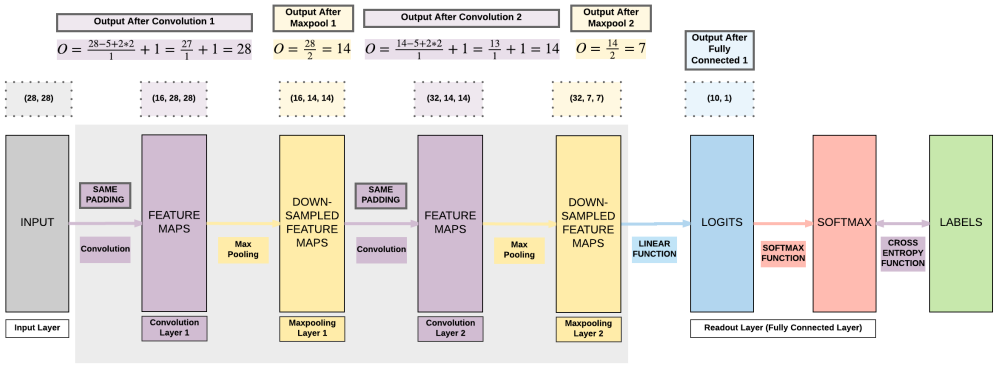


Figure 3: Pipeline of a typical convolution

target/goal , while respective steering commands (consisting of rotational and translational velocities) are the output. The network parameters are learned during training. RELU activation function is used at the two convolution layers. Max Pooling and average pooling is used in two layers. The optimization is executed according to the *Adam* optimizer with mini-batch training. The loss function for each supervised learning step k is as given in equation (2).

$$J_k(\Gamma_B) = \frac{1}{N_B} \cdot \sum_{j=i}^{i+N_B} |\mathcal{F}_{\theta_k}(y_j, g_j) - \mathbf{u}_{exp,j}| \tag{2}$$

On the global level, a grid-based Dijkstra planner is used while on the local level a DWA approach planner is used. The ROS stage 2D is used as the dynamic simulator, and the entire training is done on the simulation. This is a crucial decision in our entire pipeline.Instead of training the network on real data, it is being trained on a simulation. That is in another way also a test of the generalization and vigorousness of the network. For basic tests , the simulation eliminates the noise from the data for training. The data is shuffled randomly

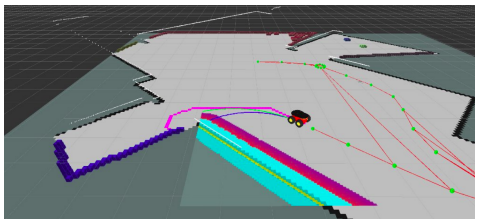


Figure 4: A typical ROS simulation

1.5 The advantages of using a DNN as Motion Planner

The biggest advantage is the forseeable query time during testing. The complexity of this approach doesn't change or increase with the complexity of the network. The feature (or training data) which is the laser data doesn't require any kind of preprocessing. It is being fed directly to the network. Hence , when the network is trained the complexity of query time becomes constant. No extra effort needs to be put for any logical programming or algorithm analysis, since it will all be taken care of by the **trained CNN** network.

1.6 Frame by Frame Evaluation

For training and evaluation purposes, three maps were generated , i.e, *train*, $eval_1$ and $eval_2$ respectively. The robot drives to randomly selected locations on *train* and the simulated sensor data, relative target positions and the steering commands (the translational component and rotational component) were stored. 6000 trajectories were generated in the *train* map with 2.1M input/output features and outputs respectively. The training of the network model was done using these features from *train* only, and that too on the `CNN_smallFC` model. For evaluation of our network, an amalgamation of the features from *train*, $eval_1$ and $eval_2$ were used . This data was unseen to the network during the training . A comparision is shown in **Fig 5**, between the ROS expert motion planner and our network during the evaluation, for the rotational and translational components respectively. As the figure shows (the lower portion), if we see *train* and $eval_1$, they are different in structure but the obstacles remain same in nature. This was a factor for the increase in rotational velocity error , and the errors further increased in case of $eval_2$.
It has been explicitly mentioned in the paper that during sharp turns, i,e *during a 180° turn* the error in rotational velocity increases drastically which is reflected in the error evaluation. The purpose of this experimentation was to prove that ***transfering knowledge from one map to another is quite possible for this network***, under the constraint that ***results were analyzed frame by frame*** in the simulation environment.

1.7 Analysis of the model in the simulated environment

Let the constraint be removed in the previous scenario, and the trained network is deployed as a motion planner. The model and the *train* data to train the network are kept constant and this particular experimentation is basically a verification of the fact if **there is any overfitting in the network after training it on a specific map**. **Fig 6** shows the experimentation for two scenarios,*train* and $eval_1$. The deep planner understands only relativistic local goals(shown by the red dots in **Fig 6**), but the expert ROS planner has knowledge about the whole map.

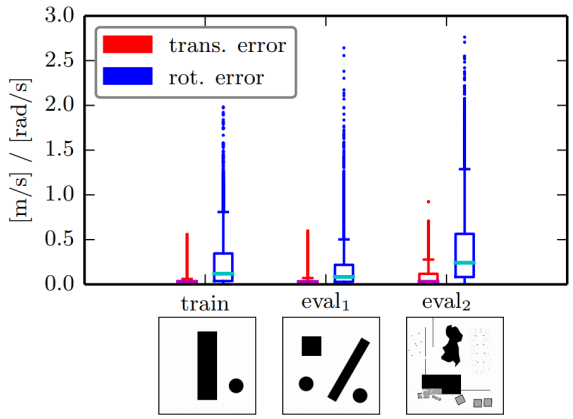


Figure 5: Error Statistics of Frame By Frame Evaluation between ROS expert and our model

The following metrics are used in the paper for evaluation of the network:
d_{goal}: distance of the final trajectory w.r.t the goal for all the trajectories.
E_{trans}: summation of all absolute values of the translational acceleration over all the trajectories.
E_{rot}: summation of all absolute values of the rotational acceleration over all the trajectories.
dist: total distance travelled for one experimental pipeline.
time: total time for one experimental pipeline.

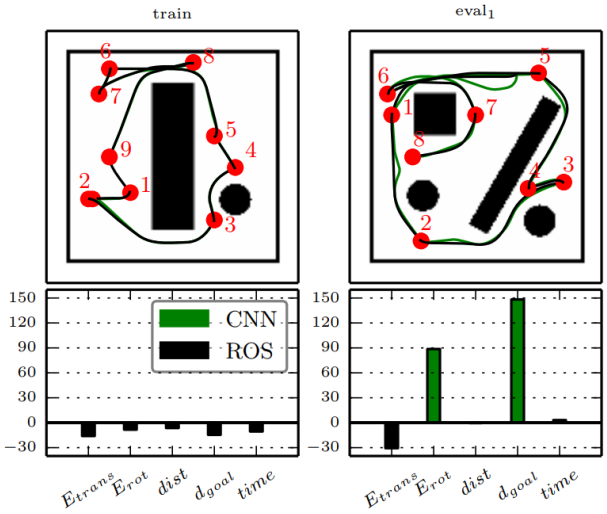


Figure 6: Performance comparision between the ROS(expert) and the deep planner

As per the comparison of trajectories shown in **Fig 6**, the trajectory on *train* performs better than on $eval_1$. But this is clearly not a case of overfitting because the network was not tuned for these metrics. These are experimental results. As far as the metrics are concerned the CNN performs it better on *train* than $eval_1$, and this is reflected in the statistics shown in the bar charts in **Fig 6**.

1.8 Testing the trained network on Real Environment

The CNN, which was trained in a simulation environment on *train*, is tested on a real environment which is similar in structure to $eval_2$. The major change which is done to the network architecture here,is to the Fully Connected Layers, and it's specifications were changed to *CNN_bigFC* model.

Target	1	2	3	4	5	6	7	8	9	10	11	12	13
CNN_smallFC	1	0	1	0	4	0	0	0	2	4	6	0	0
CNN_bigFC	0	0	1	1	0	1	0	0	0	0	4	1	0

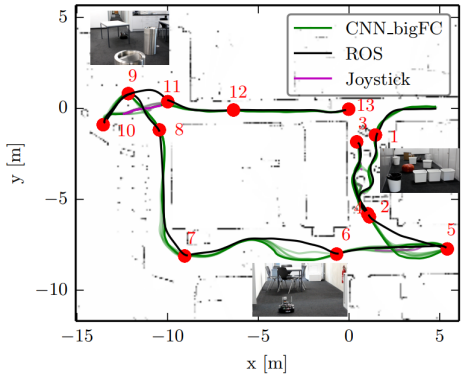


Figure 7: Comparision of trajectories between the ROS(expert) and the deep planner on a real environment.

Fig 7 shows the trajectories for this changed network (trained before on simulation). The obstacles were real world obstacles of course, because this is a real scenario, but the targets were still localized for the deep network model. This experiment was not fully autonomous. It required human intervention in the form of joystick movements to control the robot at places where it got stuck , and the trained network was unable to take any decision. The joystick interventions were recorded for both the versions of the model - *CNN_smallFC* and *CNN_bigFC*. As per the observations recorded in **Fig 7** the network with larger Fully Connected Layer gives better results.Interestingly, without the joystick the behaviour was uncomprehended but no collisions or stray movements were recorded.

Another experiment was performed in the real world setup, to see how the network performed during testing if dynamically changing obstacles appeared in its line of motion. As shown in **Fig 8** the robot changes its path as soon as the laser readings show an obstacle,but it changes its path back again when the obstacle is removed. This is a direct verification of the trained network and a proof that it can transfer its learning from a simulation environment to the real world in a dynamic scenario.

2 Discussion

The most important finding from this work is the demonstration that a navigation policy can be learned by a CNN with a residual architecture. Rather than delving into the question why specifically this architecture works, the paper focuses on experimenting with this specific architecture and trying to find a generic limitation to its capabilities. The most novel part of the experiment was in proving that this network can be trained in a simulation environment

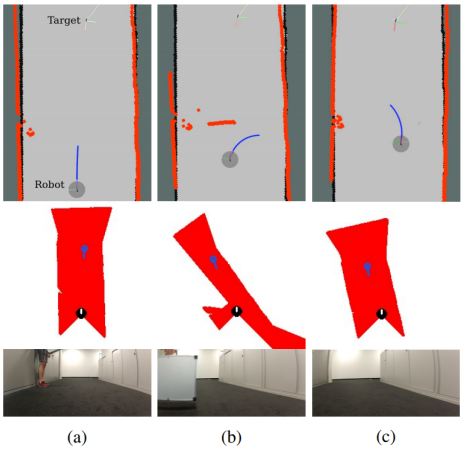


Figure 8: Behaviour of the robot when a dynamic obstacle appears in vicinity

and then the trained model can be used in a real test environment. The transfer capability of this model is indeed very efficient, but the extent to which a trained network can perform is not quite well discussed in the paper. There might be scenarios where the robot might get stuck in a local minima , as shown in **Fig 9** . In such a scenario this model might not be able to perform well. Moreover, human intervention is involved during the real world testing of the trained network on the robot, in the form of a joystick. It is used as a parameter to test the performance of the network in this experiment as per **Fig 7** but this is not acceptable in a practical environment.

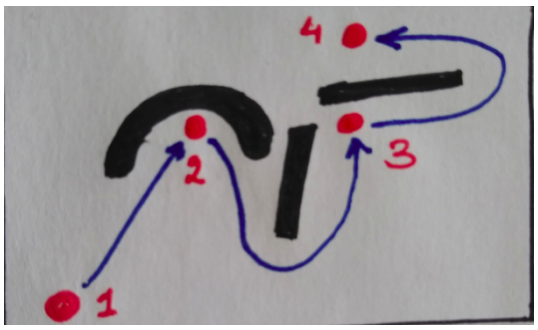
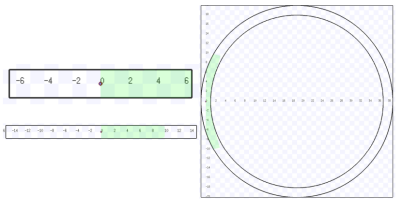


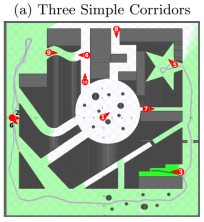
Figure 9: A scenraio when there might be a local minima

But there have been other researches for the exact same problem statement. The most prominent one is using multimodal auto-encoder instead of a resNet as in [2]. The architecture of this network consists of a combination of Restricted Boltzmann machines(RBMs) and deep auto-encoder.

Fig 10 shows some more complicated scenarios tested using the auto-encoder approach in [2]. Another way of solving this problem is using a basic MLP as in [3]. This research group has actually implemented a path finding algorithm using neural networks, which is a motion planner for a robot and they have also solved it on a map as shown in **Fig 11**,which is quite similar in nature to our problem. They have applied pre-processing on the dataset obtained from the sensors, and they used PCA to extract meaningful features from the dataset and then train the network using the feature with reduced dimensional. Pre-processing of the



(a) Three Simple Corridors



(b) Complex Environment

Figure 10: Scenarios tested in [2]

dataset is something which was not explored in our paper before feeding it in the CNN.

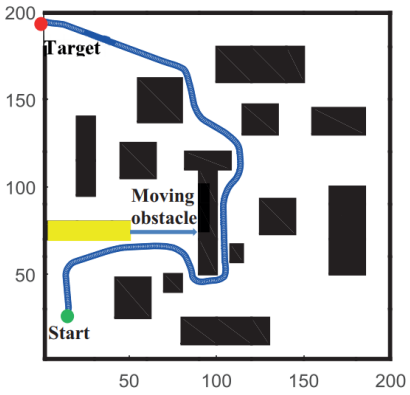


Figure 11: Scenarios tested in [3]

Fig 12 shows the algorithm and their network, which is discussed in [3].

Another novel approach has been done for a similar motion planning problem using MP-Nets in [4], on both 2D and 3D environments. But the scope of this dataset is much more as it involves pointclouds rather than a data from laser sensor. By combining the CNN with a RNN, a special type of network has been created specifically for path planning in [5], which has been termed as **Value Iteration Network**. This **VIN** or **MPNet** can be modified to fit our problem statement, and trained on a simulation and tested on a real environment, with similar dataset. Although such experiments have not been performed, there is no reason why these networks won't give comparable results to our CNN discussed till now.

3 Conclusion

The approach discussed in this paper is one of the many possible methods which can be used for experimenting with motion planning using neural networks. The CNN (resNet) architecture has been defined in details and it is trained and tested on the claimed data-set comprising laser sensor and goals. The authors of this paper claims that this is the first

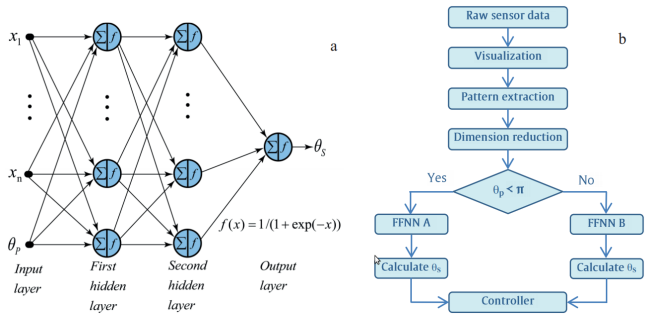


Figure 12: Neural network and Flowchart as discussed in [3]

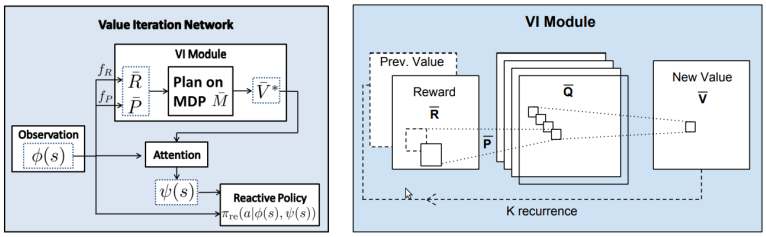


Figure 13: A typical VIN as discussed in [5]

approach which performs goal oriented motion planning, incorporating collision avoidance. The other major point is the ability of the network to transfer the learning from a simulation and execute it on a real environment. The authors have accepted that this network is unable to replicate an exact motion planner, and it does have limitations. Complex environments cannot be traced out by this network since it relies heavily on local planning. Dead-ends cannot be figured out in these experiments without human interruption. Obstacles which cannot be reflected from laser sensors (like glass) cannot be handled properly by the robot because the training was done in a simulation environment. Better methods do exist compared to this technique which involve RNNs, preprocessing of the data using dimensionality reduction and autoencoders which have been discussed in previous sections.The other approach to solve this genre of problems is using reinforcement learning and deep reinforcement learning, which can be much more effective for motion planning in an environment with dynamic obstacles.

References

[1] Kaiming He, Xiangyu Zhang ,Shaoqing Ren ,Jian Sun [*Microsoft Research*].
Deep Residual Learning for Image Recognition.

[2] James Sergeant, Niko Sunderhauf, Michael Milford, Ben Upcroft.
[Multimodal Deep Autoencoders for Control of a Mobile Robot]
ARC Centre of Excellence for Robotic Vision.
Queensland University of Technology, Australia

[3] Farhad SHAMSFAKHR, Bahram SADEGHIBIGHAM,
[A neural network approach to navigation of a mobile robot and obstacle avoidance in dynamic and unknown environments]

Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences, Zanjan, Iran

[4] Ahmed H. Qureshi, Anthony Simeonov, Mayur J. Bency and Michael C. Yip.
[*Motion Planning Networks*].

[5] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel.
Value Iteration Networks
[*Dept. of Electrical Engineering and Computer Sciences, UC Berkeley*].