A REPORT

ON

Mobile Robot Path Planning Using Deep Learning Techniques

BY

ABHIGYAN GANDHI **2018A7PS0168U** **CS**

AT

BITS, Pilani – Dubai Campus
Dubai International Academic City (DIAC)
Dubai, U.A.E

Second Semester, 2021-2022

A REPORT

ON

Mobile Robot Path Planning Using Deep Learning Techniques

BY

ABHIGYAN GANDHI 2018A7PS0168U CS

Prepared in Fulfillment of the
Project Course: CS F376

AT

BITS, Pilani – Dubai Campus
Dubai International Academic City (DIAC)
Dubai, UAE

Second Semester, 2021-22

BITS, Pilani – Dubai Campus
Dubai International Academic City (DIAC)
Dubai, UAE

Course Name: Design Project

Course No: CS F376

Duration: 4 Months

Date of Start: 29.01.2022

Date of Submission: 27.05.2022

Title of Report: Mobile Robot Path Planning Using Deep Learning Techniques

Name: Abhigyan Gandhi

ID Number: 2018A7PS0168U

Discipline: Computer Science

Name of Project Supervisor: Dr. V. Kalaichelvi

Keywords: Path planning, deep learning techniques, GAN, CNN, tensorflow.

Abstract:
The main objective of this project is to apply deep learning techniques for path planning of the Turtle Bot using the acquired datasets. For problems involving path planning for mobile robots, it is possible to utilize neural networks to enable to the robot to perceive the environment and perform feature extraction. Path planning of mobile robots is performed to check motion of the robot with different trajectories and analyze the patterns. We used a GAN architecture for data augmentation and used the results to train and test a CNN architecture for image classification. The datasets were acquired manually in the grid environment where the turtle bot will maneuver. GANs are used for data augmentation as our local dataset is small in size. We will do this using the python language using the PyCharm IDE.

Signature of the Student
Date: 27.05.2022

Signature of Faculty
Date: 27.05.2022

ACKNOWLEDGEMENTS

Firstly, I would like to express my heartfelt gratitude to the Director of BITS Pilani, Dubai Campus, Prof. Srinivasan Madapusi, who has ushered a new light on our college.

Next, I would like to thank Dr. V. Kalaichelvi, my Project Supervisor, for her faith in me even before I started working under her and for her motivation, support, guidance, and encouragement throughout the course of this project. She has been a great influence, urging me to learn and to be innovative, pushing me to excel.

I would also like to thank Dr. Jagdish Nayak, HOD of the Electrical and Electronics Department, for this great opportunity to pursue my Lab Project in the first semester of the academic year 2021-2022.

Lastly, I would like to thank Dr. Ram Karthikeyan and Mrs. Abhilasha (Ph.D. Scholar), for their additional support and help regarding my project.

Abhigyan Gandhi
2018A7PS0168U

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
PROBLEM FORMULATION

1.1 INTRODUCTION

Path planning for mobile robots is a field where a lot of work has been done in the recent years. Researchers have come up with numerous techniques for increasing the accuracy. The target is to create an optimal path connecting the start and end points in an environment keeping in mind the dynamics on the robot, time taken and collisions. Deep learning techniques have given a new perspective to the study, and we will be using them for our project.

1.2 OBJECTIVES

This project focuses on acquiring datasets for the robot using an external webcam, pre-processing the data, and using it for image classification and path planning. First we will divide the dataset into four parts, front, back, right, and left for the four directions it can move in. Then we will perform data augmentation on the datasets using a generative adversarial network architecture which will help in increase the dataset and also to get useful information about the data. We will use a convolutional neural network to detect objects which were places in the path of the robot while taking the datasets. Then for comparative analysis we will also implement a hybrid CNN-LSTM model for image classification. We will implement the architectures using the Tensorflow machine learning library in the PyCharm IDE for python development. This library provides us with a number of different functions for processing the data and easy use of the architectures.

1.3 PROPOSED METHODOLOGY

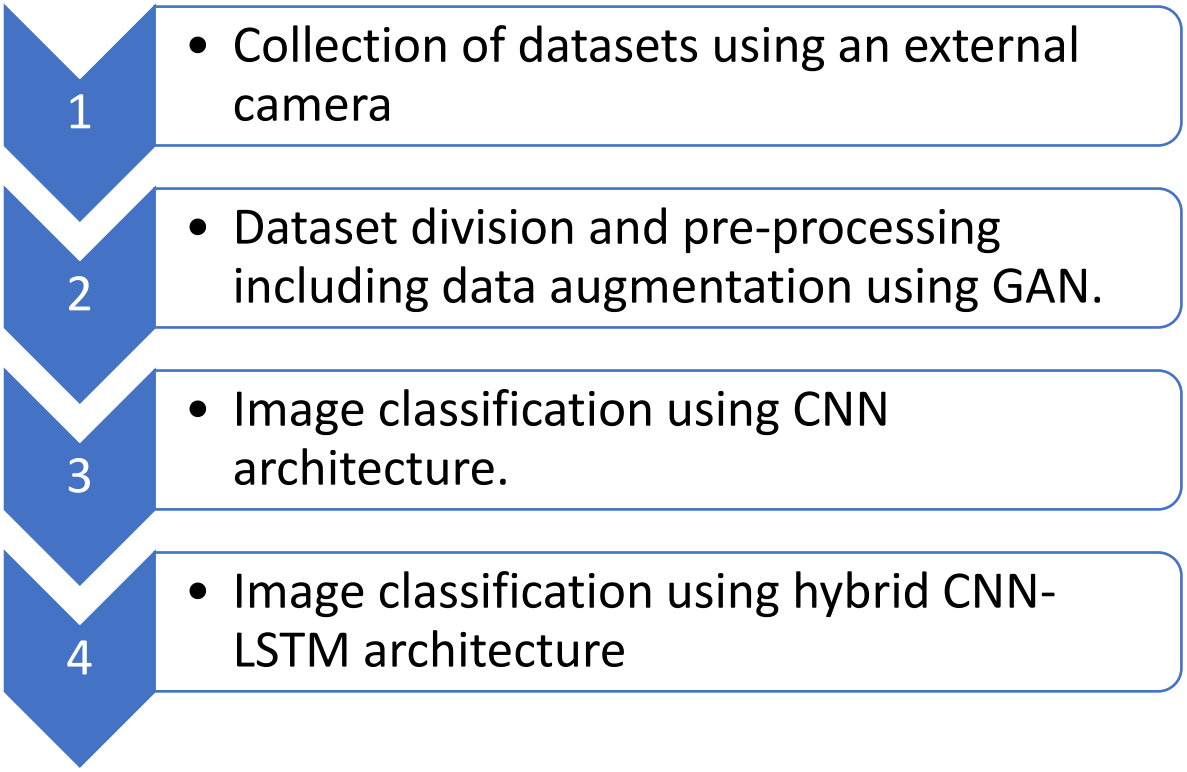


Figure 1: A flowchart of proposed methodology

CHAPTER 2
LITERATURE SURVEY

2.1 LITERATURE SURVEY PAPERS

In [1], the authors have improved the traditional RRT* algorithm by combining it with a Conditional Generative Adversarial Network (CGAN) to find an optimal path efficiently. THE CGAN produces a possibility distribution of the all the paths which can be used by the RRT* to find the best one by using a non-uniform sampling strategy.

In [2], the authors have used a CNN model which helps in extracting data feature maps from the images and compare it to the topological maps of the environment. They also introduced a new loss function for smaller datasets which takes into account the input when it associated to its true class as well other classes.

In [3], the authors have developed a Convolutional Neural Network Trajectory Tracker (CNNTT), to control the non-linear kinematics of the robot. They used multiple algorithms with it including the Chaotic Particle Swarm Optimization (CPSO), A-star algorithm, and the Hybrid Swarm Optimization (HSO) algorithm. These were implemented using MATLAB and the results they got from CPSO and HSO were accurate and gave the shortest paths.

In [4], the authors have developed a novel image-based path planning algorithm by using a GAN which takes in the map of the environment and for the output gives a map where the promising area is segmented (feasible path possible). They also used non-uniform sampling as a heuristic for path planning. Upon analysis they found that this strategy is much better than uniform sampling in terms of quality and convergence speed.

In [5], the authors have improved the CaffeNet architecture by using making to some of the layers. It is basically a CNN which has been used for path planning. They developed their datasets manually by taking images of their local environment. The changes they made resulted in improved performance in terms of time and memory.

In [6], the authors have given a deep look into deep reinforcement learning techniques. They talk about algorithms based on value functions and strategy gradient in path planning. They talk about the future of deep reinforcement learning in the field of path planning and works on improving the algorithm for real-world use.

CHAPTER 3
IMAGE CLASSIFICATION FOR PATH PLANNING

3.1 EXISTING ALGORITHMS FOR PATH PLANNING

Research in the field of path planning of mobile robots has become an important subject as the usage of mobile robots has increased massively in a number of different sectors. Automation of tasks which are both easy and hard to carry out has been the goal for industries as it is less time consuming and requires lesser human effort. Optimization of the path of mobile robots plays a huge factor as it necessary for efficiency and researchers have developed a number of intelligent systems and algorithms for it. Some popular algorithms include the RRT algorithm, neural network-based algorithms, potential field method and many more.

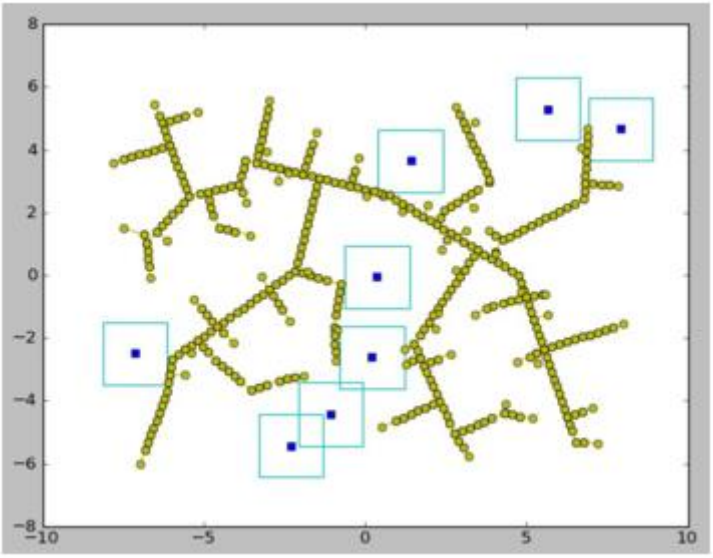


Figure 2: RRT Algorithm Graph

There are two types of path planning systems based on the environment, global and local path planning. Global path planning can be done on a global map in which you only know the shape and position of the obstacles. The environment needs to be figured out and then the algorithms are used in that global map for path planning. Local path planning focuses on the local environment and takes information from it using the robot's sensors to define an optimal path. This method is based on adjusting the path simultaneously while learning about the environment and figuring out the path.

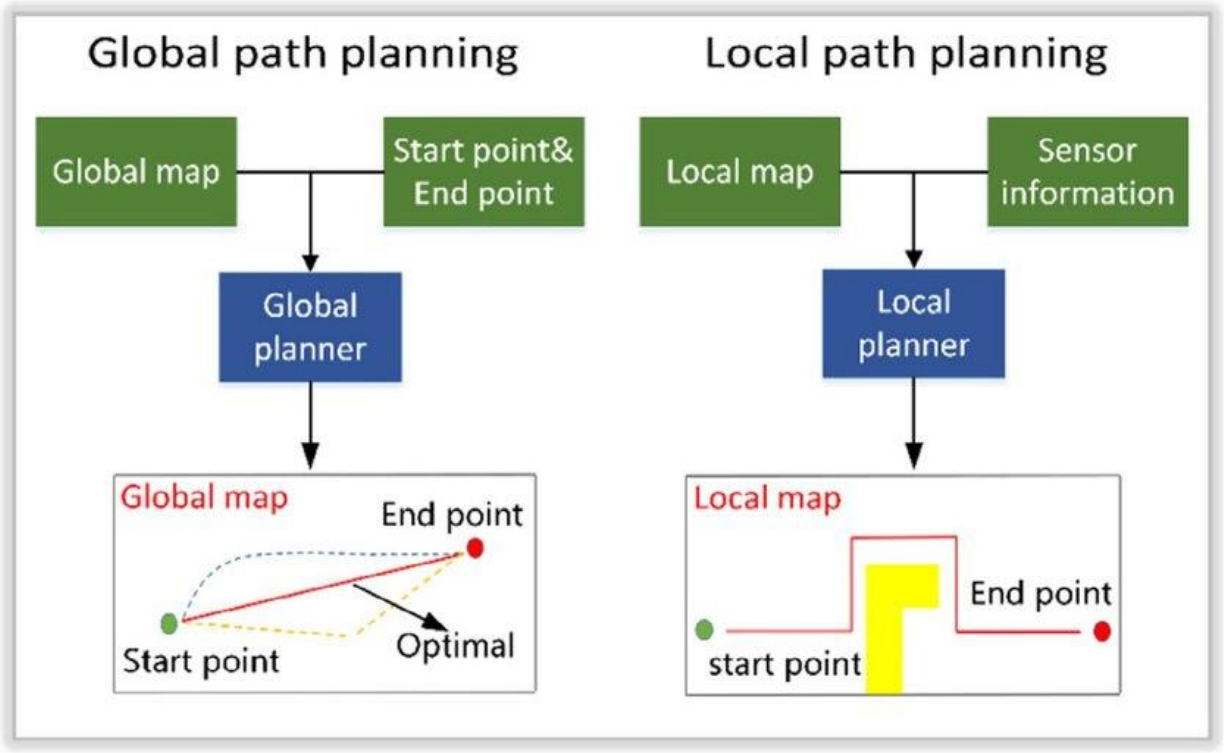


Figure 3: Global and Local path planning

Various types of path planning algorithms are used in real world situations, one of the prominently used algorithm is the sampling-based path planning. This algorithm basically creates a graph connecting points which have no obstacles. It checks if there are any obstacles in the path and if not then it adds them to a sample and creates a map from them. This algorithm is used majorly in disassembling and assembling problems in which an optimal path has to be found for taking apart or putting together an object while taking into consideration the feasibility and maintenance for the system. Real time problems which need constant checking of collisions like a dual crane lifting problem use a parallel algorithm which keeps on checking the paths of the cranes and detects any collision before time and prevent it. Path planning algorithms can also be used in GPS systems by using the preplanned routes of vehicles present in the system which can help in minimizing the traffic.
Different algorithms also come with some drawbacks like the neural network and the particle swarm algorithms require high computational power and data. The A* and potential field method can sometimes produce redundant output which slows down the process. Overcomplicating the system which needs more memory can be taxing and can lead to less optimal results.

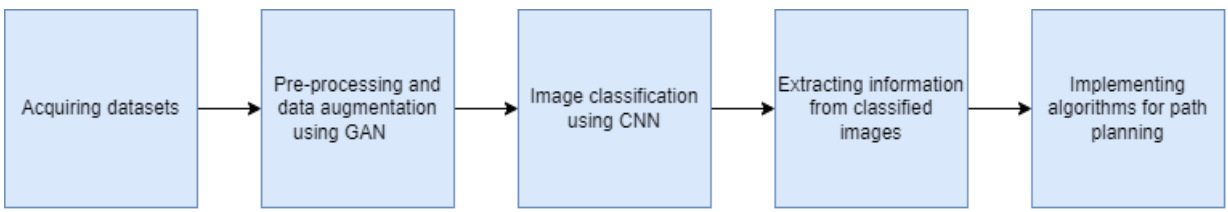


Figure 4: Block diagram of project

3.2 DATA AUGMENTATION USING GANS

For the case of mobile robot path planning in local environments, there can be issues with datasets regarding their size and features. To overcome these issues, we can use data augmentation techniques. In recent years, the use of generative adversarial networks has increased massively.
Generative Adversarial Networks are a model which were developed for generating new samples from the input which are exactly as the input. They are most commonly implemented using convolutional neural networks and are an unsupervised learning method which means that no prior output is available for reference and the features are learned from the inputs. The model is built to differentiate the real and generated images to learn better.
GANs consist of two parts: a generator function and a discriminator function.
The generator is used to create new data samples by taking input from the original dataset and then the real and generated images are sent to the discriminator to distinguish between real and fake images.
The model can be considered as a competition between the generator and the discriminator, in which both the functions try to beat the other. The generated and real images are sent to the discriminator which produces binary output which in turn helps in developing the generator to produce more real-like images.

3.3 DATA ACQUISITION

For the datasets, we manually took images of the environment where the turtle bot will move. The environment is a grid of size (200, 400).

We also took images from the perspective of the robot's camera by placing the webcam at the level of the robot.

Figure 5: External camera used

A VGA webcam was used for taking the images of 640x480 resolution. It is a USB camera which can easily be connected to your device. To take the pictures we used a simple script using the OpenCV library in python. We kept the camera on the turtle bot at the height of its camera and then while manually rotating it to cover different angles captured the pictures and saved them on our device.

TurtleBot is a ROS standard platform robot. For our project we have used the TurtleBot 2i. The TurtleBot 2i uses advanced modules which can be combined with 3D cameras. It has two cameras, one for its movement and one for the robot arm.

Figure 6: TurtleBot 2i

TurtleBot 2i ROS Software/Demo Features

- Autonomous Navigation & Point Cloud Mapping
- Map Zone Identification & Waypoints
- Robotic Arm Object Manipulation & Sorting
- Obstacle Avoidance & Path Planning
- Teleoperation Examples
- Self-Charging w/ Dock

Hardware specifications:

CPU:

- Intel Joule 570X
- Gumstix Nodana Carrier Board
- 4GB RAM
- 16GB eMMC Storage
- 802.11AC WiFi / Bluetooth 4.0
- Ubuntu 16.04 / ROS Kinetic

TurtleBot 2i Sensors

- SR300 RealSense 3D Camera
- ZR300 RealSense 3D Camera
- Accelerometer/Gyro/Compass
- Edge Detection & Bumper Sensors

TurtleBot 2i Mobile Robot

- Kobuki Mobile Base
- Modular & Interchangeable Decks
- Pincher MK3 Robo Arm
- Arbotix-M Robocontroller
- Secondary 3S 4500mAh LiPo Battery
- Maximum translational velocity: 70 cm/s 13
- Maximum rotational velocity: 180 deg/s (>110 deg/s gyro performance will degrade)
- Payload: 2kg (without arm), 1kg (with arm)
- Cliff: will not drive off a cliff with a depth greater than 5cm
- Threshold Climbing: climbs thresholds of 12 mm or lower
- Rug Climbing: climbs rugs of 12 mm or lower
- Expected Operating Time: 4-6 hours (operating time varies depending on loadout)
- Expected Charging Time: 2-3 hours (charge time varies depending on loadout)
- Docking/Charging: automatic within a 2mx5m area in front of the docking station (sold separately)

The dataset is divided into four parts, front, back, left, and right according to the directions the robot will move in after detecting the object in front of it.

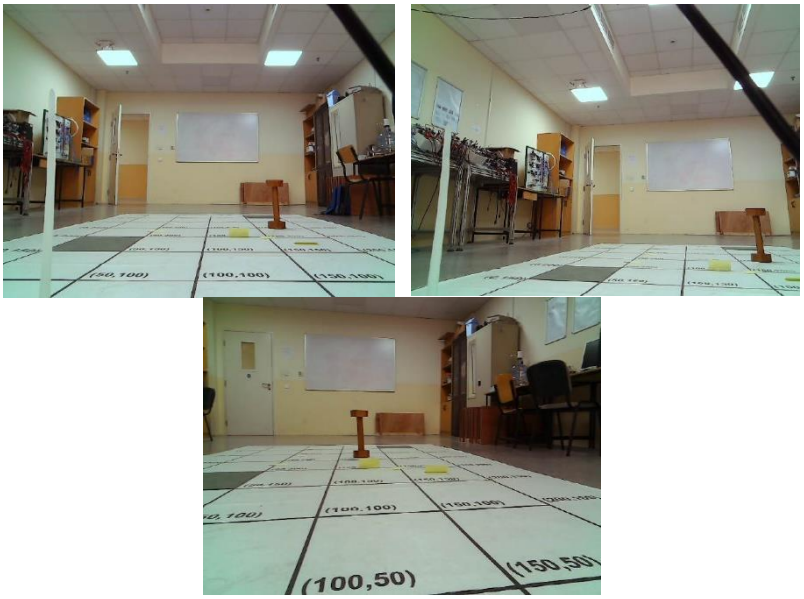


Figure 7: Sample of front images

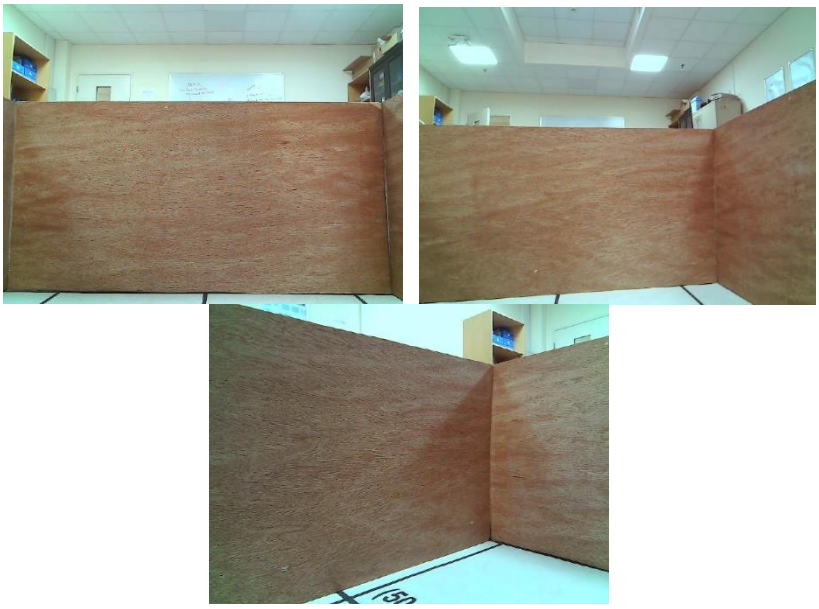


Figure 8: Sample of back images

Figure 9: Sample of right images

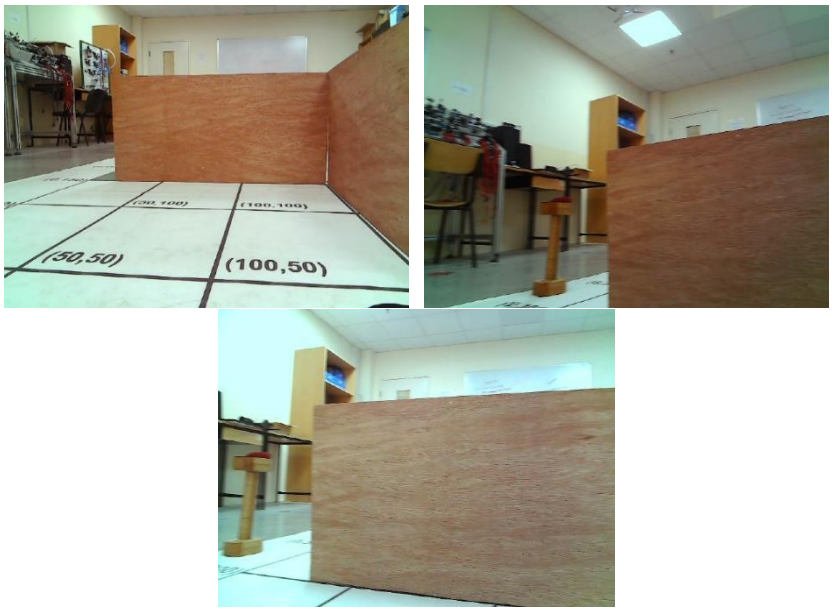


Figure 10: Sample of left images

CHAPTER 4
DEEP LEARNING ARCHITECTURES USED FOR PATH CLASSIFICATION

4.1 INTRODUCTION

In this section we will give details about the GAN architecture of our project.
The functions we have implemented include a generator, a discriminator, a generator loss function, a discriminator loss function, a train step, and train function and a generate and save image function.

4.1.1 GENERATOR

It consists of a dense layer, a reshape layer, 6 convolutional layers with batch normalization and ReLU layers.

The dense is just a layer of neurons which are defined as per the need. Multiple dense layers can be used together to make a deeply connected network.

The Reshape layer is used to reshape the inputs into the given shape.

The batch normalization layer is used to normalize the inputs by maintaining the mean output and standard deviation around the values 0 and 1, respectively. While the model is in the training phase the mean and standard deviation are adjusted based on the current sample batch of the inputs and while testing a moving average of the values is taken from the seen input batch.

The leaky ReLU layer is a leaky version of a rectified linear unit. It allows a small gradient when the unit is not active.
f(x) = alpha * x if x < 0
f(x) = x if x >= 0

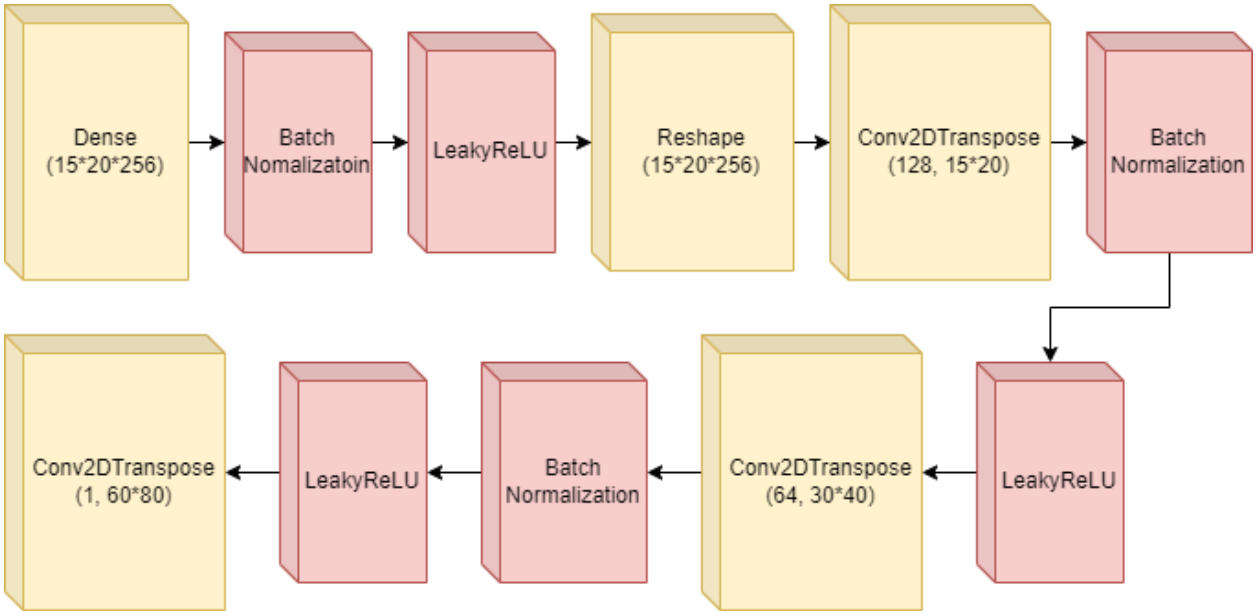


Figure 11: Generator architecture

The Conv2DTranspose layer is used to use a transformation going in the opposite direction of a normal convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution. It has a number of arguments. The ones we have used include: 1.filters: gives the dimensionality of the output space. 2.Kernel_size: defines the height and width of the 2D convolution window. 3.Strides: Specifies the strides of the convolution along the height and width. 4.Padding: we have used 'same' padding which results in padding with zeros evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input. 5.Activation: we have used the 'tanh' activation function.

4.1.2 DISCRIMINATOR

It consists of two Conv2DTranspose layers, a flatten layer, a dense layer with leaky ReLU and dropout layers.

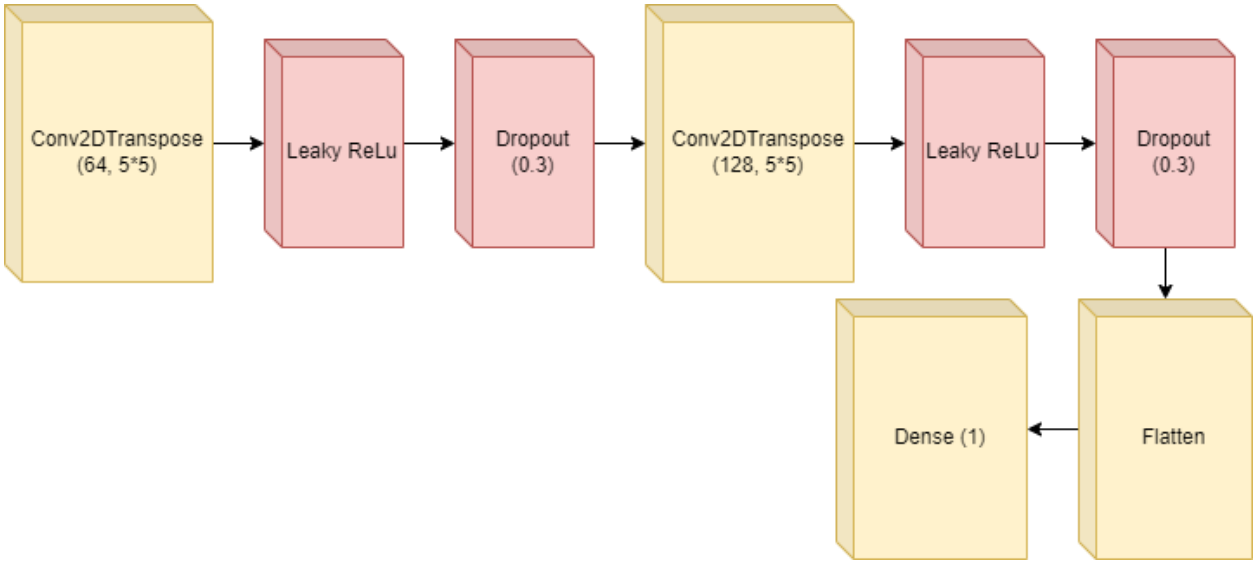


Figure 12: Discriminator architecture

The Conv2DTranspose layers, Dense layer and Leaky ReLU layers perform the same function as the generator.

The dropout layer is used to prevent overfitting of the model, which basically means that it stops the model from accurately mapping all the training samples as if it does that then it will not perform well on new unseen test data. The layer randomly sets the defined percent of the inputs to 0, and to keep the sum total same it increases some of the other inputs by the ratio of 1 is to 1 – rate.

The flatten layer flattens the input without affecting the batch size.

4.2 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks have brought a major change in the image classification and object detection field. They have helped in advancements with computer vision and deep learning. CNNs are a machine learning algorithm best suited for anything to do with images. They are created in such a way that reading images and their pixel values is very easy for the network. In mathematics, convolution basically means the modification of shapes with respect to another shape which is basically what CNNs do. They are a deeply connected high neuron network and have the property of learning and updating themselves rather than having to fix the filters and variables beforehand. You can have as many layers in a CNN as you want. The first layer is for understanding the less complex features like colors and edges. Then more layers can be added to get more complex information from the images which will help in getting a feasible output. When an image is inputted to a CNN layer, a convolutional window needs to be defined which are basically the pixel values of how big you want the window to be. Then the layer will start to move that box from top left to the right bottom throughout the image while understanding the features.

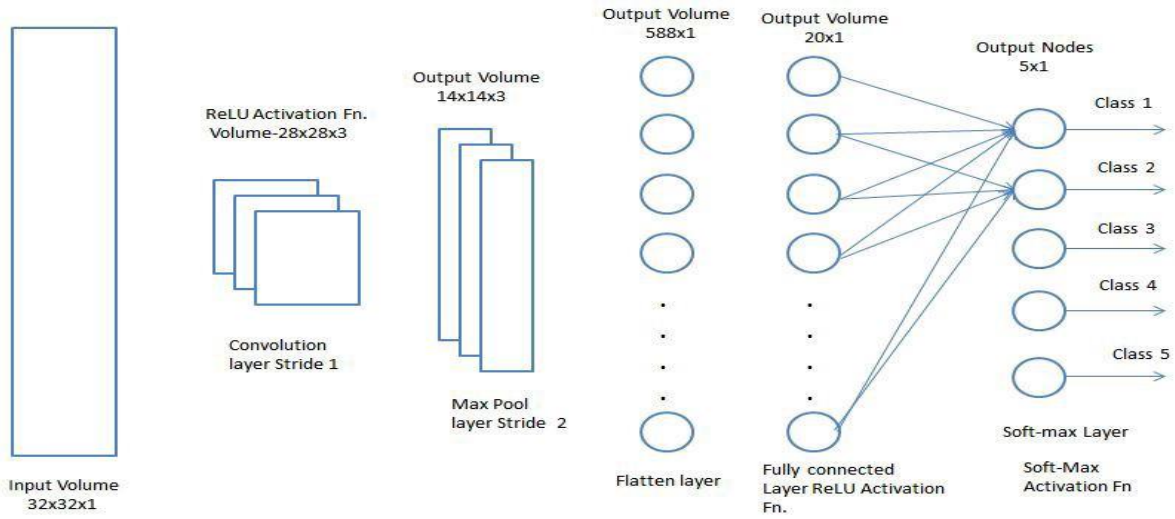


Figure 13: CNN general model

In our architecture, there are three Conv2D layers with MaxPool2D layers. Followed by a flatten layer to convert input from 2D to 1D, which goes into the three fully connected dense layers.

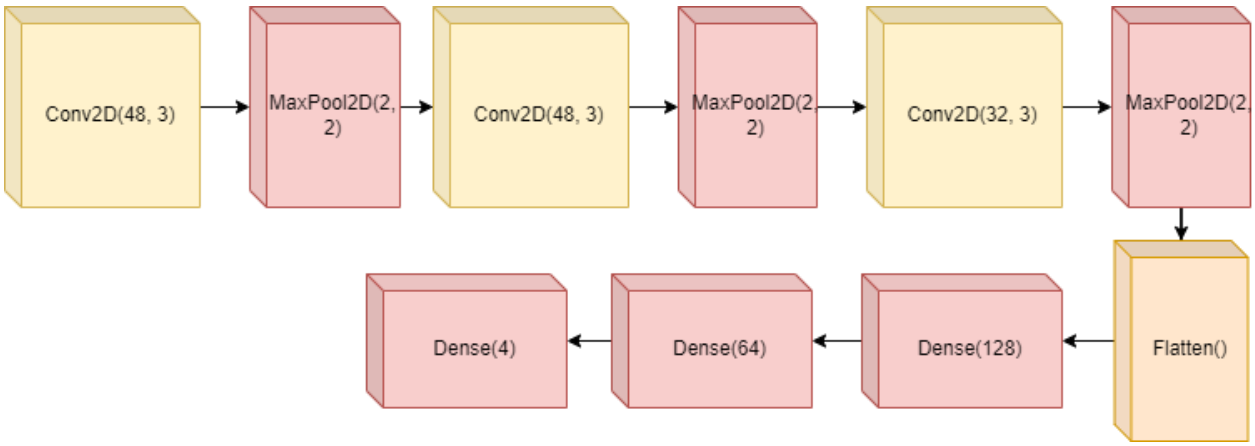


Figure 14: CNN architecture

4.3 HYBRID CNN-LSTM NETWORKS

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex problem domains like machine translation, speech recognition, and more. They can be combined with convolutional neural networks and used to perform image classification. The convolutional layers are used as in normal CNN models, with the addition of LSTM layers before the dense output nodes. This model was implemented using the keras sequential model, by using the tensorflow.keras.layers.LSTM function. In the convolutional layers the TimeDistributed function can be added so that their output can be used with the LSTM layers.
The model implemented in this paper includes four convolutional layers with MaxPool2D layers, a flatten layer, two LSTM layers, followed by three dense layers.

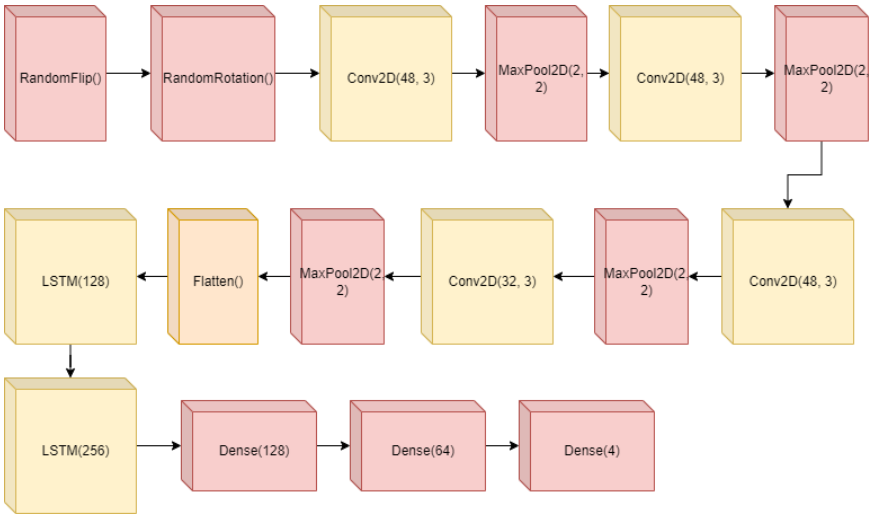


Figure 15: CNN-LSTM architecture

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 WORK DONE AND RESULTS

First we reviewed literature papers revolving on the topic of path planning using deep learning techniques. We provided an overview of work researchers have accomplished and took inspiration from that. Then we learned about GANs and CNNs and how to use them for path planning. We created manual datasets by clicking photos of the environment of the robot and pre-processed them. We used data augmentation techniques to increase the size of our dataset. Implemented a CNN image classification model.
For image classification, we used the keras sequential model which is a neural network class having multiple uses. Using the Conv2D layer, we implemented a CNN architecture. For model compilation, we used the Adam optimizer with categorical_crossentropy as the loss function. Categorical_crossentropy is used for multi-class classification, which is the case we are working with, having four classes, front, back, left, and right. For the performance metrics we used, accuracy, precision, recall and IoU.
Following are the details about the CNN model:

Model summary:

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 60, 80, 3)	0
conv2d (Conv2D)	(None, 60, 80, 48)	1344
max_pooling2d (MaxPooling2D)	(None, 30, 40, 48)	0
conv2d_1 (Conv2D)	(None, 28, 38, 48)	20784
max_pooling2d_1 (MaxPooling2D)	(None, 14, 19, 48)	0
conv2d_2 (Conv2D)	(None, 12, 17, 48)	20784
max_pooling2d_2 (MaxPooling2D)	(None, 6, 8, 48)	0
conv2d_3 (Conv2D)	(None, 4, 6, 32)	13856
max_pooling2d_3 (MaxPooling2D)	(None, 2, 3, 32)	0
flatten (Flatten)	(None, 192)	0
dense (Dense)	(None, 128)	24704
dense_1 (Dense)	(None, 64)	8256
dense_2 (Dense)	(None, 4)	260

Total params: 89,988
Trainable params: 89,988
Non-trainable params: 0

Figure 16: CNN model summary

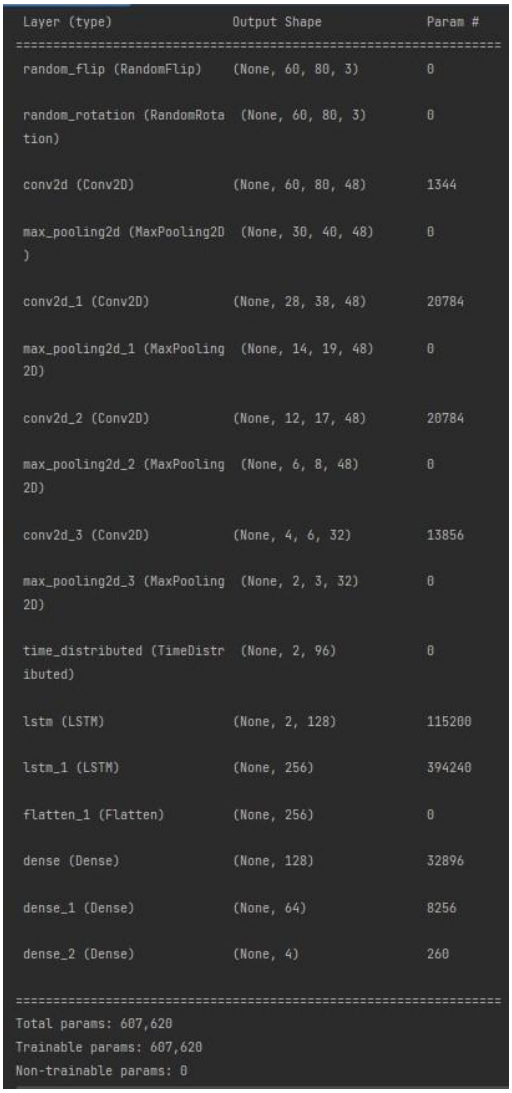
CNN model Hyperparameters:

Table 1: CNN model hyperparameters

Parameter	Value
Optimizer	Adam
Learning rate	$3*10^{-4}$
Epochs	30
Batch size	32
Pool method	Max
Loss	Categorical cross-entropy
Padding	Same

Following are the details about the CNN-LSTM model:

Model summary:



```
Layer (type)                 Output Shape              Param #
=================================================================
random_flip (RandomFlip)     (None, 60, 80, 3)         0

random_rotation (RandomRota  (None, 60, 80, 3)         0
tion)

conv2d (Conv2D)              (None, 60, 80, 48)        1344

max_pooling2d (MaxPooling2D  (None, 30, 40, 48)        0
)

conv2d_1 (Conv2D)            (None, 28, 38, 48)        20784

max_pooling2d_1 (MaxPooling  (None, 14, 19, 48)        0
2D)

conv2d_2 (Conv2D)            (None, 12, 17, 48)        20784

max_pooling2d_2 (MaxPooling  (None, 6, 8, 48)          0
2D)

conv2d_3 (Conv2D)            (None, 4, 6, 32)          13856

max_pooling2d_3 (MaxPooling  (None, 2, 3, 32)          0
2D)

time_distributed (TimeDistr  (None, 2, 96)             0
ibuted)

lstm (LSTM)                  (None, 2, 128)            115200

lstm_1 (LSTM)                (None, 256)               394240

flatten_1 (Flatten)          (None, 256)               0

dense (Dense)                (None, 128)               32896

dense_1 (Dense)              (None, 64)                8256

dense_2 (Dense)              (None, 4)                 260

=================================================================
Total params: 607,620
Trainable params: 607,620
Non-trainable params: 0
```

Figure 17: CNN-LSTM model summary

CNN-LSTM model hyperparameters:

Table 2: CNN-LSTM hyperparameters

Parameter	Value
Optimizer	Adam
Learning rate	$3*10^{-4}$
Epochs	30
Batch size	32
Pool method	Max
Loss	Categorical cross-entropy
Padding	Same
Dropout	0.3
Recurrent dropout	0.2

Categorical cross-entropy is used in multi-label classification and gives output that only one of the classes out of all is correct. Padding can be of two types valid and same. Same padding is used here as it makes sure that the output size is same as the input size. The dropout parameter helps to stop overfitting, it has a value of 0.3 which means that it drops 30% of the batch input of 0 before proceeding to the next layer in the network.

After running both the models the following results were obtained:

Table 3: Results

Metric	CNN	CNN-LSTM
Training accuracy	97%	82%
Validation accuracy	86%	82%
Precision	83.7%	81.4%
Recall	82.4%	80.6%
IoU	0.65	0.55

The precision and recall values help us define the quality and quantity of the network. Precision is the ratio of true positive labels to the true positive plus false positive labels. Recall is the ratio of true positive labels to the true positive and false negative labels.

High values of 80-84% of both the metrics tell us that most of the labels are being predicted correctly.

From the results of the two models, we can see that the CNN model performs better while training and the validation outputs are similar. This difference is because in the

CNN-LSTM model, the convolutional layers were sent through the TimeDistributed function which applies a layer to every temporal slice of the input, it converts the 2D input to 3D format which is necessary for classification using the LSTM layers. This disturbs the input, and the training accuracy decreases.

Sample output images of CNN model:

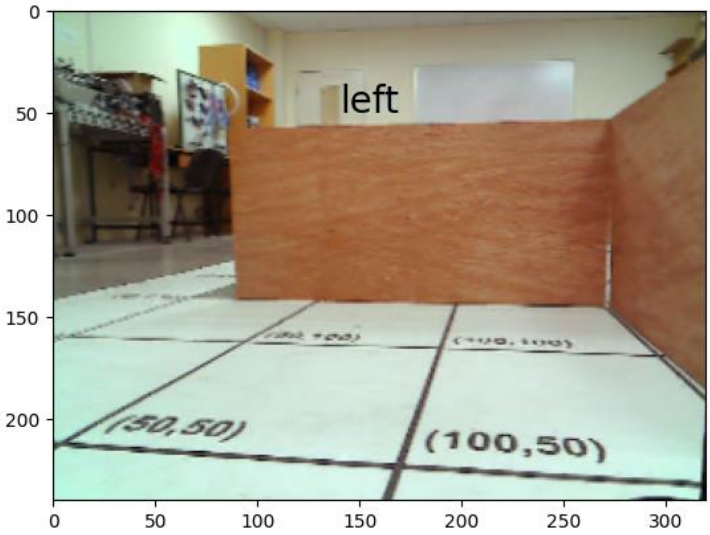


Figure18: Left classified image

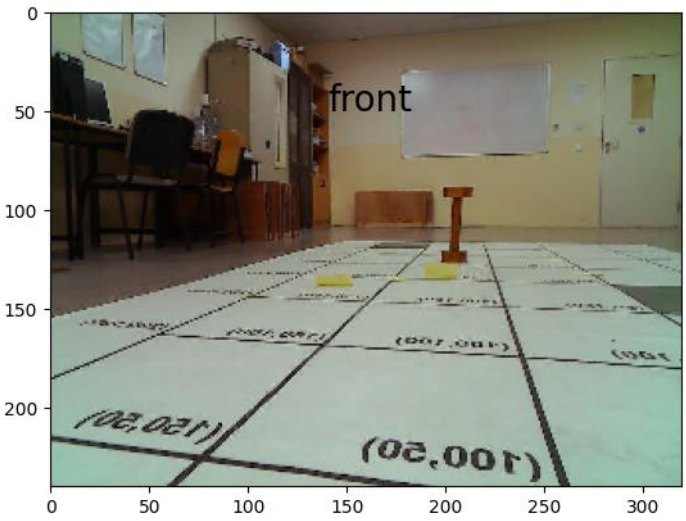


Figure 19: Front classified image

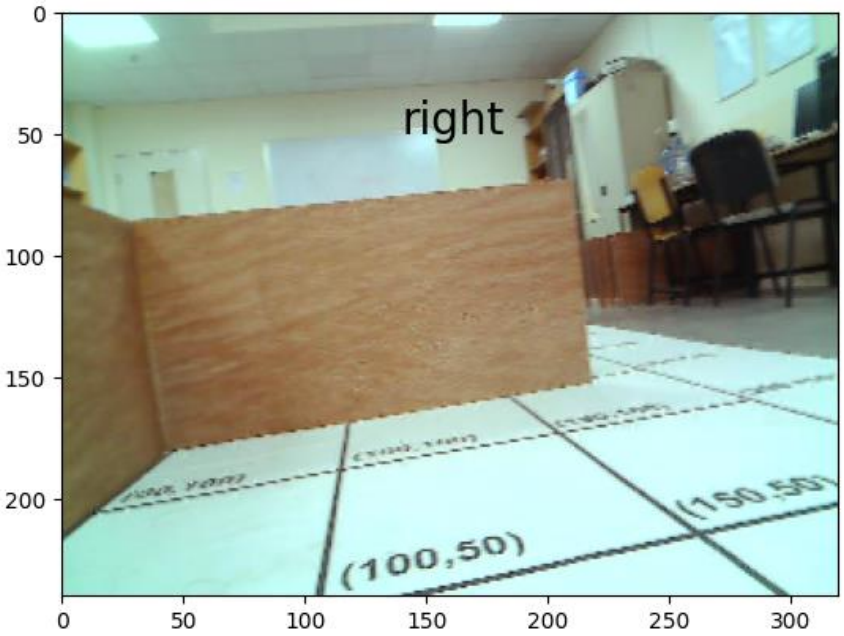


Figure 20: Right classified image

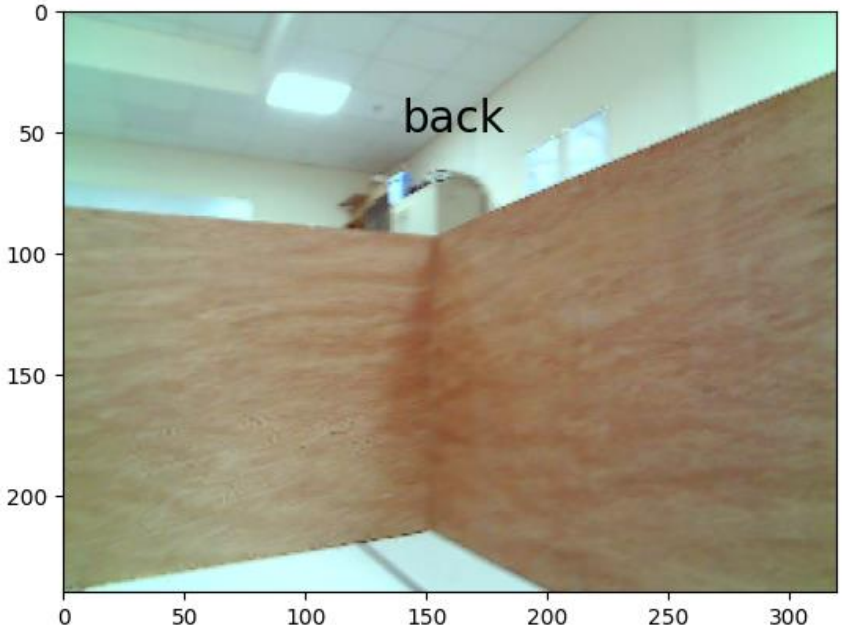


Figure 21: Back classified image

Sample output images of CNN-LSTM model:

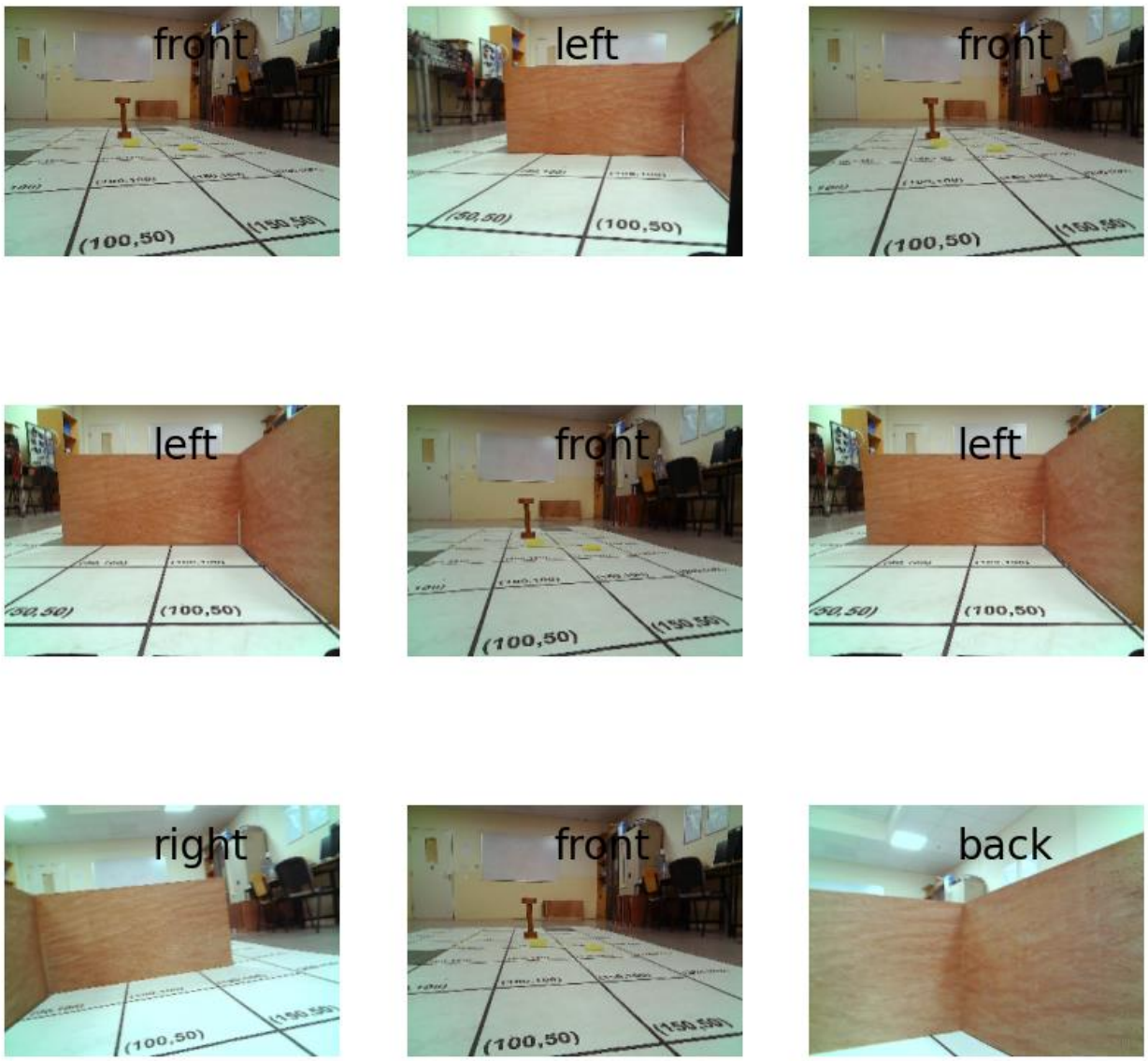


Figure 22: CNN-LSTM classified output images

`The GAN model was run for 5000 epochs and started to learn and develop features from the input images.

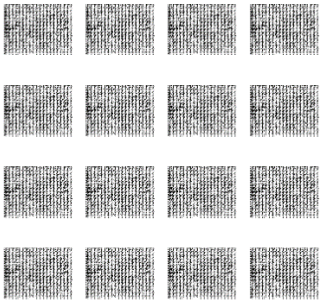


Figure 23: GAN output after 1000 epochs

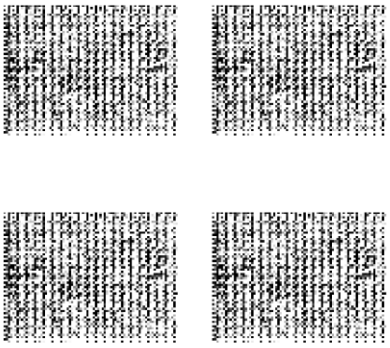


Figure 24: Gan output after 2000 epochs

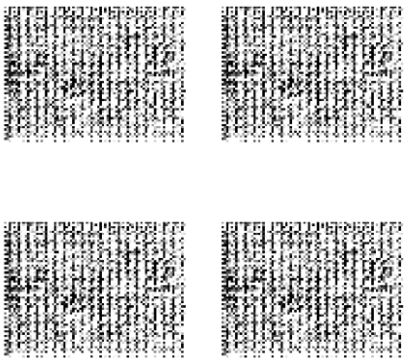


Figure 25: GAN output after 3000 epochs

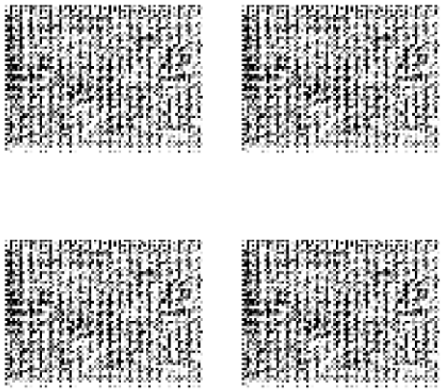


Figure 26: Gan output after 4000 epochs

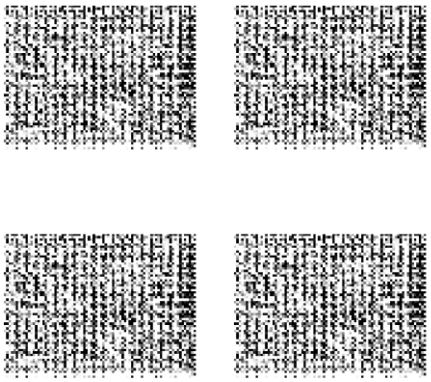


Figure 27: GAN output after 5000 epochs

5.2 CONCLUSION

In this report, we performed image classification for path planning of mobile robots in a local environment. The datasets were manually acquired, pre-processed and data augmentation was performed using a GAN architecture. For image classification, two models were used for comparative analysis, namely, a CNN model and a hybrid CNN-LSTM model. On performance analysis, it was seen that the CNN model performed better while training giving am accuracy of 97% while the CNN-LSTM model had 86%. The validation accuracy of the both the models was in the same range of 82%. The GAN model had some limitations but after running it for around 5,000 epochs, it started to learn features from the input images and produce feasible output images.

5.3 FUTURE WORK

To further improve this model, changes to the GAN model can be done. The generator model was not strong enough to generate images similar to the input images even after 5,000 epochs, this can be improved by adding more convolutional layers to the network and reduce the size of input images. The number of neurons can also be increased to the network denser. Further we a more complicated GAN loss function can be used which takes into consideration more factors. For the CNN mode, the accuracy can be improved by increasing the size of the validation dataset and adding more data augmentation layers.

REFERENCES

[1]N. Ma, J. Wang, J. Liu, and M. Meng, "Conditional Generative Adversarial Networks for Optimal Path Planning", *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1-9, 2021. Available: 10.1109/tcds.2021.3063273.

[2]F. Foroughi, Z. Chen and J. Wang, "A CNN-Based System for Mobile Robot Navigation in Indoor Environments via Visual Localization with a Small Dataset", *World Electric Vehicle Journal*, vol. 12, no. 3, p. 134, 2021. Available: 10.3390/wevj12030134.

[3]O. Wahhab and A. Al-Araji, "Path Planning and Control Strategy Design for Mobile Robot Based on Hybrid Swarm Optimization Algorithm", *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 3, pp. 565-579, 2021. Available: 10.22266/ijies2021.0630.48.

[4]T. Zhang, J. Wang, and M. Meng, "Generative Adversarial Network Based Heuristics for Sampling-Based Path Planning", *IEEE/CAA Journal of Automatics Sonica*, vol. 9, no. 1, pp. 64-74, 2022. Available: 10.1109/jas.2021.1004275.

[5]A. Shirbandi, F. Saberi and M. Rostami, "Path Planning Algorithm for Mobile Robot based on Deep Learning", *2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pp. 1-7, 2021. Available: 10.1109/icrom54204.2021.9663444.

[6]Y. Zhao, Y. Zhang, and S. Wang, "A Review of Mobile Robot Path Planning Based on Deep Reinforcement Learning Algorithm", *Journal of Physics: Conference Series*, vol. 2138, no. 1, p. 012011, 2021. Available: 10.1088/1742-6596/2138/1/012011.

[7] Y. Chen, U. Rosolia and A. Ames, "Decentralized Task and Path Planning for Multi-Robot Systems", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4337-4344, 2021. Available: 10.1109/lra.2021.3068103.

[8] T. Zhang, J. Wang, and M. Meng, "Generative Adversarial Network Based Heuristics for Sampling-Based Path Planning", *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 1, pp. 64-74, 2022. Available: 10.1109/jas.2021.1004275.

[9] Z. Li, J. Wang, and M. Meng, "Efficient Heuristic Generation for Robot Path Planning with Recurrent Generative Model", *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1-7, 2021. Available: 10.1109/icra48506.2021.9561472.

[10] J. Yuan, H. Wang, C. Lin, D. Liu and D. Yu, "A Novel GRU-RNN Network Model for Dynamic Path Planning of Mobile Robot", *IEEE Access*, vol. 7, pp. 15140-15151, 2019. Available: 10.1109/access.2019.2894626 [Accessed 8 April 2022].

[11]N. Guo, C. Li, T. Gao, G. Liu, Y. Li and D. Wang, "A Fusion Method of Local Path Planning for Mobile Robots Based on LSTM Neural Network and Reinforcement Learning", *Mathematical Problems in Engineering*, vol. 2021, pp. 1-21, 2021. Available: 10.1155/2021/5524232 [Accessed 8 April 2022].

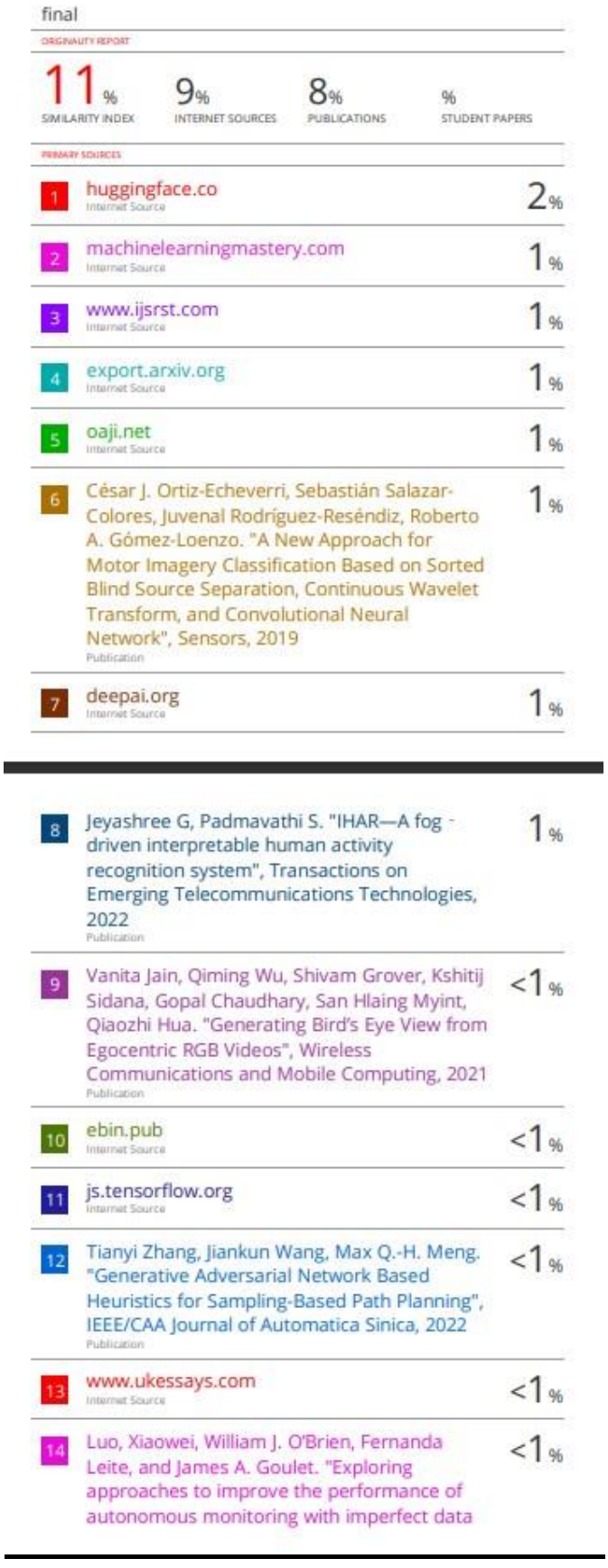
[12]Y. Lu, W. Wang, and L. Xue, "A Hybrid CNN-LSTM Architecture for Path Planning of Mobile Robots in Unknow Environments", *2020 Chinese Control and Decision Conference (CCDC)*, 2020. Available: 10.1109/ccdc49329.2020.9164775 [Accessed 8 April 2022].

[13]H. Zhang and M. Li, "Rapid path planning algorithm for mobile robot in dynamic environment", *Advances in Mechanical Engineering*, vol. 9, no. 12, p. 168781401774740, 2017. Available: 10.1177/1687814017747400 [Accessed 8 April 2022].

[14]Q. Yao et al., "Path Planning Method with Improved Artificial Potential Field—A Reinforcement Learning Perspective", *IEEE Access*, vol. 8, pp. 135513-135523, 2020. Available: 10.1109/access.2020.3011211 [Accessed 16 April 2022].

[15]B. Yang, L. Wellhausen, T. Miki, M. Liu, and M. Hutter, "Real-time Optimal Navigation Planning Using Learned Motion Costs", *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. Available: 10.1109/icra48506.2021.9561861 [Accessed 16 April 2022].

TURNITIN REPORT



final

ORIGINALITY REPORT

11%	9%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	huggingface.co Internet Source	2%
2	machinelearningmastery.com Internet Source	1%
3	www.ijsrst.com Internet Source	1%
4	export.arxiv.org Internet Source	1%
5	oaji.net Internet Source	1%
6	César J. Ortiz-Echeverri, Sebastián Salazar-Colores, Juvenal Rodríguez-Reséndiz, Roberto A. Gómez-Loenzo. "A New Approach for Motor Imagery Classification Based on Sorted Blind Source Separation, Continuous Wavelet Transform, and Convolutional Neural Network", Sensors, 2019 Publication	1%
7	deepai.org Internet Source	1%
8	Jeyashree G, Padmavathi S. "IHAR—A fog - driven interpretable human activity recognition system", Transactions on Emerging Telecommunications Technologies, 2022 Publication	1%
9	Vanita Jain, Qiming Wu, Shivam Grover, Kshitij Sidana, Gopal Chaudhary, San Hlaing Myint, Qiaozhi Hua. "Generating Bird's Eye View from Egocentric RGB Videos", Wireless Communications and Mobile Computing, 2021 Publication	<1%
10	ebin.pub Internet Source	<1%
11	js.tensorflow.org Internet Source	<1%
12	Tianyi Zhang, Jiankun Wang, Max Q.-H. Meng. "Generative Adversarial Network Based Heuristics for Sampling-Based Path Planning", IEEE/CAA Journal of Automatica Sinica, 2022 Publication	<1%
13	www.ukessays.com Internet Source	<1%
14	Luo, Xiaowei, William J. O'Brien, Fernanda Leite, and James A. Goulet. "Exploring approaches to improve the performance of autonomous monitoring with imperfect data	<1%

in location-aware wireless sensor networks", Advanced Engineering Informatics, 2014.
Publication

15 Lorenz Wellhausen, Marco Hutter. "Rough Terrain Navigation for Legged Robots using Reachability Planning and Template Learning", 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021 — <1%
Publication

16 cyberleninka.org — <1%
Internet Source

17 portfolios.cs.earlham.edu — <1%
Internet Source

18 www.hindawi.com — <1%
Internet Source

19 Haoxuan Huang, Xiaoyu Zhang, Ziyu Zhao. "COVID-19 Detection Based on 2D Parallel CNN Model", 2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI), 2021 — <1%
Publication

Exclude quotes Off
Exclude bibliography On
Exclude matches Off