Environment Identification

Rahul Shah
Università Della Svizzera Italiana
Lugano, Switzerland
shahra@usi.ch

Rahul J Hirur
Università Della Svizzera Italiana
Lugano, Switzerland
rahulh@usi.ch

***Abstract*—In earlier times, there was not any model proposed for making prediction of the room but in the era of new technology we are able to develop a model which can be implimented to any robot which can detect room on basis of surrounding environment using CNN (i.e. Convolutional Neural Network) also based on probability of image fed to CNN. We are building the similar model using the MobileNetV2 for prediction of room using MyT.**

I. Problem Statement :

Consider a robot that might be in one of N places (e.g. inside a class, in the courtyard, in a corridor; or in a simulated room vs a different simulated room). You want to build a place recognition CNN that given an image frame returns in which place the robot is. Design a controller such that, when the robot is placed in one place, it acquires many images of that place. Use the controller to collect a dataset, train a CNN and show that it works (e.g. the robot could light up its lights in different ways)

II. Objectives :

In pursuit of addressing the outlined problem statement, the following objectives are delineated to guide the development and evaluation of a place recognition CNN for the robotic system

- Develop a place recognition CNN capable of accurately determining the robot's location within a predefined set of places.
- Design and implement a controller mechanism that facilitates the acquisition of multiple image frames by the robot when positioned in a specific place.
- Curate a comprehensive dataset utilizing the controller to capture diverse visual representations of each distinct location.
- Train the CNN model using the acquired dataset to enable robust place recognition capabilities.
- Validate the efficacy of the trained model through practical demonstrations, such as having the robot manipulate its lighting system in varying configurations corresponding to recognized places.

III. Proposed Solution :

Based on a detailed observation of the problem statement, we have formulated a comprehensive solution that involves the following six steps:

1) Retrieve initial dataset.
2) Develop a CNN to classify the images into "N" Classes(such as Rooms, corridor etc.).
3) Gather Information(Images) from the environment the robot is placed in by exploring around.
4) Classify the robot environment.
5) In case of lower confidence, move to another place in environment and re-run the detection algorithm.
6) After successful classification, making it visible to viewer
   a) Light up the robot which indicates the specific environment.
   b) Save the image into the dataset(Optional).

IV. Architecture :

Our implementation is generally based on python script and ROS (Robot Operating System) an evironment which is standard facto for robot programming .Also, coppeli-asim where we can simulate the robot.

A. Architecture consist of

- **MyT:** Diffrential Robot where we used 5 proximity sensor , camera sensor , light in project as shown in Fig [1]
- **ROS :** It is environemt where we can do robot programming with python API script.
- **Coppelia-sim :** Where we can simulate robot by creating environment as in ".ttm" format.
- **Convolutional Neural Network :** Its artificial neural network which is created using Pytorch framework.

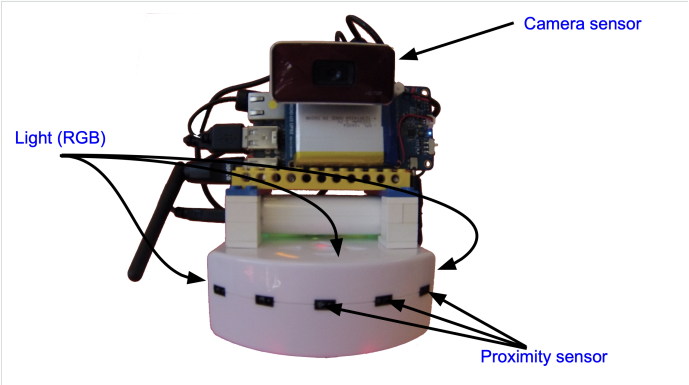

Fig. 1. MyT

V. Kick start :

As we detailed in I we need to create an CNN which will detect the room and based on probabilities of images taken from environment we need to predict room or environment. So, to make it work we have created an environment in coppelia-sim in which we have created two rooms identified with

- **Room _1** : Decorated with different objects and even contradiction in floor color (such as Robots and cupboard) as shown in Fig [2] we gave label for this room as Robotic Garage.
- **Room _2** : Decorated with different objects and even contradiction in floor color (such as people, tables, plants) as shown in Fig [3] we gave label for this room as Waiting room.

Now, for acquiring images we have used MyT camera sensor and of-course, for exploring . Furthermore, for making predictions we have created Convolutional Neural Network (in our case we have used MobileNet _v2) which trains the images and makes predictions.

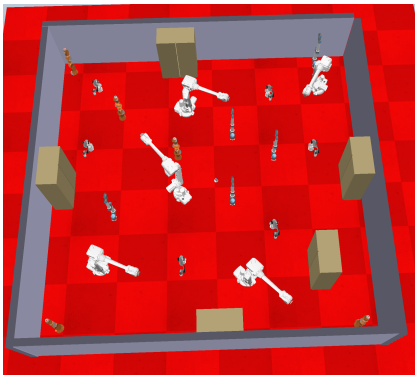

Fig. 2. Room _1

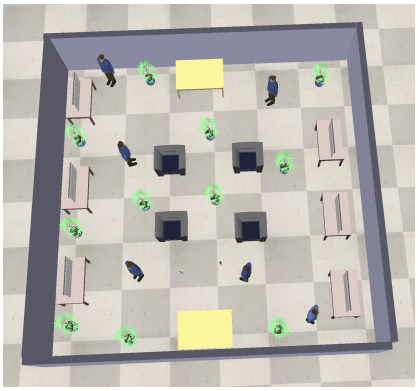

Fig. 3. Room _2

VI. Implementation :

We used MyT to explore rooms one by one as mentioned above in Fig 2 and Fig 3 and with exploring using OpenCV images are taken from environment and those are train in CNN to predict the room and glow lights as room predicted.

Detail architecture :

A. Random exploring :

Here,we have assigned robot under free space environment where it explores by randomly exploring . Detail analysis for this algorithm is defined below as in form of states and its transitions also shown in Fig 4

STATES :

- STATE-1 : [Forward] robot moves straight ahead
- STATE-2 : [Back-up] The robot moves backward to avoid an obstacle
- STATE-3 : [Rotating] The robot rotates in place to change its direction.

Transition of STATES :

- Forward to Back-up : Occurs when any of the front sensors detect an obstacle closer than ”Target distance”.
- Back-up to Rotating : Occurs when all the front sensors detect no obstacle within ”Too close” distance.
- Rotating to Forward : Occurs when all the front sensors detect no obstacles within their range.

The schematic view is shown below

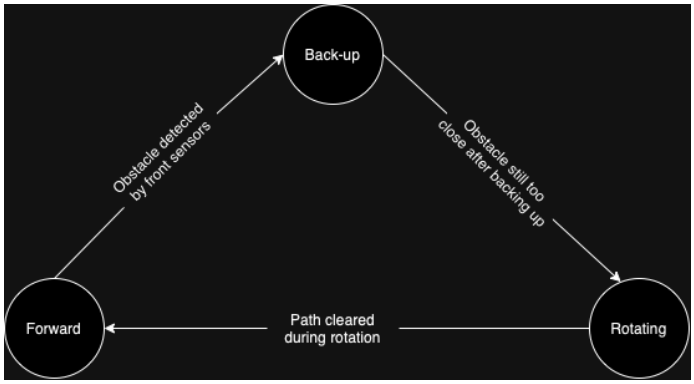

Fig. 4. State Transition

B. Data Interpretation :

In order to facilitate the training of our Convolutional Neural Network (CNN) model, we have undertaken the acquisition of images from the camera sensor of MyT-Thymio. Recognizing the importance of ample data for robust model performance, we have meticulously captured approximately 450 images per room, with a total of two rooms under consideration. Each image acquisition occurs at regular intervals of 5 seconds.

C. Convolutional Neural Network:

Now, we have enough data to train our CNN model, but there arises a question: what if there are more similar or irrelevant images, which may create distractions during prediction? To avoid this, we define a function that discards images whose variance is less than a threshold, while the rest are classified into training and test datasets for both rooms. Additionally, we have defined two models:

1) **Model-1**

- Under this model, we have used MobileNetV2, which is a pre-trained deep neural network specifically used for image classification.
  Architecture:
  a) Initial Convolution (1 layer): A 3x3 convolutional layer with a stride of 2 and 32 output channels, applied to the input image to extract features.
  b) Bottleneck Blocks (51 layers):
  – A series of 19 inverted residual blocks, each consisting of convolutional layers, depthwise convolutions, pointwise convolutions, and batch normalization layers.
  – Each block increases the complexity of features and reduces spatial dimensions.
  – Expansion factor and number of output channels vary within each block.
  c) Final Convolution (1 layer): A 1x1 convolutional layer with 1280 output channels, applied to the feature map output by the bottleneck blocks to further transform the features.
  d) Fully Connected Layer (1 layer): A linear layer mapping the output of the final convolution to the desired number of classes for classification. Often followed by a softmax activation function for obtaining class probabilities.

2) **Model-2**

- Here, we have defined our own model customized respectively according to our data.
  Architecture:
  – Total of 5 layers deep
  – Two Convolutional layers
  – One Max pooling layer
  – Two Fully connected layers

Note: The models are trained with Batch _size of 32 for each of model and ran over 20 epochs with accuracy and F1 score.

D. Algorithm for prediction :

Description or steps for algorithm :

1) Logging and Model Loading:
   - Logs the start of the prediction process.
   - Loads the model if it is not already loaded.
2) Image Processing and Prediction:
   - Checks if an image file is specified.
   - Opens the image, converts it to RGB, and preprocesses it.
   - Converts the image to a tensor and prepares it for the model.
   - Uses the model to predict the image's class, computing probabilities via softmax.
3) Result Logging and LED Control:
   - Logs the predicted class and its probability.
   - If the prediction confidence is high (probability $>$ 0.5), it logs the corresponding room and activates an LED light with the color assigned to the predicted class.

Algorithm 1: Image Prediction Pseudocode

Input: Image file
Output: Predicted room and LED activation

```
while predict_image do
    Log "predicting image";
    if model is not loaded then
        Load model;
    end
    if model is loaded then
        Log "model loaded confirmed";
        if image file is provided then
            Log "Inside prediction chain";
            Open the image and convert to RGB;
            Preprocess the image;
            Convert image to tensor and add batch dimension;
            Disable gradient computation;
            Get model output for the image;
            Compute probabilities using softmax;
            Get the predicted class;
            Log predicted class and its type;
            Get probability of the predicted class;
            Log probability and corresponding room number;
            if predicted class is 0 and probability > 0.5 then
                Log "In room 1";
                Turn on LED with color for room 1;
            end
            else if predicted class is 1 and probability > 0.5 then
                Log "In room 2";
                Turn on LED with color for room 2;
            end
        end
    end
end
```

E. Performance evaluation:

For model-1:

1) Evaluations :
   - Accuracy: 96.73 %
   - Precision: 0.9680
   - Recall: 0.9673
   - F1 Score: 0.9673
2) This all performance will be used to evaluate ROC curve and confusion matrix as shown in Fig 5 and Fig 6

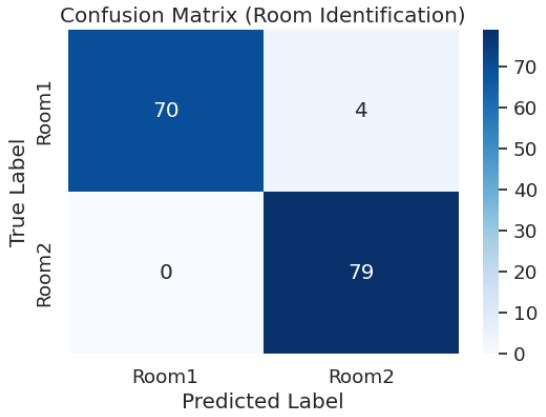

Fig. 5. Confusion matrix

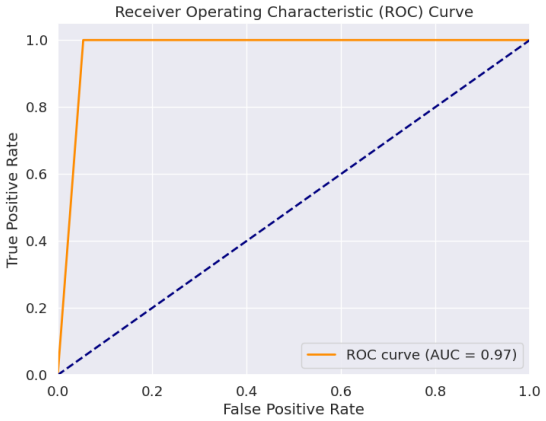

Fig. 6. ROC Curve

For model-2:

1) Evaluations :
   - Accuracy: 98.69 %
   - Precision: 0.9873
   - Recall: 0.9869
   - F1 Score: 0.9869
2) This all performance will be used to evaluate ROC curve nad confusion matrix as shown in Fig 7 and Fig 8

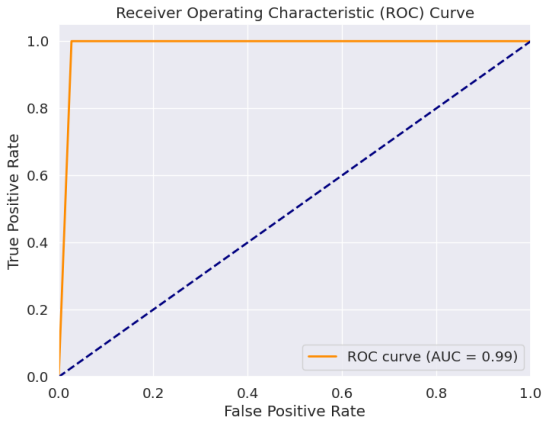

Fig. 7. ROC Curve

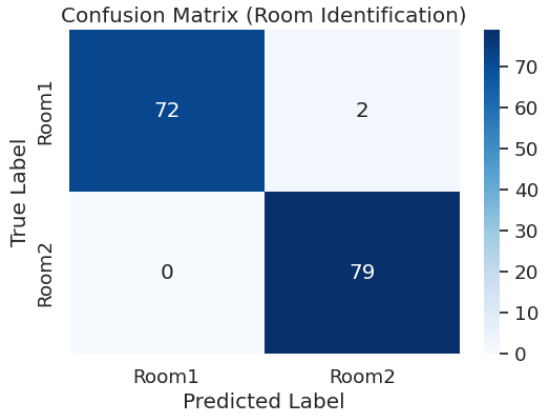

Fig. 8. Confusion matrix

VII. Conclusion :

- Lets wind-up this above results and we conclude that the model with an AUC of 0.99 is marginally better in distinguishing between the positive and negative classes also a slightly better performance with a higher true positive rate for a given false positive rate compared to the model with an AUC of 0.97. [AUC : Area Under the Curve]
- From the confusion matrices, we can see that Model-2 outperforms Model-1 specifically in identifying Room1, with more true positives and fewer false negatives.
- For Room2, both models perform equally well so both predict Room 2.
- Finally, we conclude that our model-2 is more precise than model-1 based on factors of ROC curve, Confusion matrix, Accuracy.

References

[1] Reference forMyT-Thymio configurations
[2] Reference for ROS2 documentation
[3] Reference for Pytorch documentation.
[4] Reference for sk-learn library.