Learning to drive by Crashing*

Pushkar Jajoria
Università della Svizzera italiana, Lugano
Lugano, Switzerland
pushkar.jajoria@usi.ch

Sanchit Chiplunkar
Università della Svizzera italiana,Lugano
Lugano, Switzerland
sanchit.chiplunkar@usi.ch

***Abstract*—We define and address the problem of Learning to drive MyT Thymio without using sensor values. Our main aim in this project is to drive Thymio robot using camera frame as an input instead of the traditional use of sensors as was done in previous assignment to detect and avoid any upcoming obstacles. Thus for solving this problem we have used CNN Regression model to analyse visual images and use them to predict our angular velocity and hence avoid any upcoming obstacles.**

***Index Terms*—Convolutional Neural Network,Deep Learning, ROS, MyT Thymio,PyTorch.**

I. Introduction

Almost all of human movement is based on the raw image input collected from our eyes. We are able to maneuver ourselves in complex environments with ease, only based on the inputs from our eyes. To train robots so that they can replicate the movements of humans, If not better, using only the camera frames as an input will be an ultimate achievement for robot navigation using Artificial Intelligence. In this project we have trained the MyT thymio robot to avoid and maneuver across obstacles by using only its front camera frame as an input. We did this by training a CNN regressor which predicts the angular velocity for MyT. The name in the title is derived from the data collection process, where the robot is allowed to repeatedly crash and respawn at random locations, all the while recording the sensor values. We trained a CNN regressor which leveraged upon this dataset to predict the angular velocity, derived from the sensor values. The results were compared with an augmented ResNet18 model to justify the efficacy of our approach.

II. Method

A. Data Collection

At the initial stages of data-collection, we collected image frames of size 470 $\times$ 350 $\times$ 3. The increased size showed no major improvements over a resized image of 100 $\times$ 100 $\times$ 3 at minimizing the loss. The first major improvement from the data engineering perspective was to bias the data set to collect more data near obstacles. This is evident from 1 which shows the highly skewed data set before this improvement. The dataset finally consists of 100 $\times$ 100 RGB images and their corresponding target vector of 5 sensor values from left to right. This target vector of 1x5 was finally converted to a single value of either angular velocity of a centrality by multiplying it with an averaging vector.
Another major improvement from the perspective of data engineering was to change the maximum sensor range from .12 to .75 because of the big size of our arena. This greatly helped us in getting a better model. We collected 10,000 images for the train set and 2000 images for the test set.

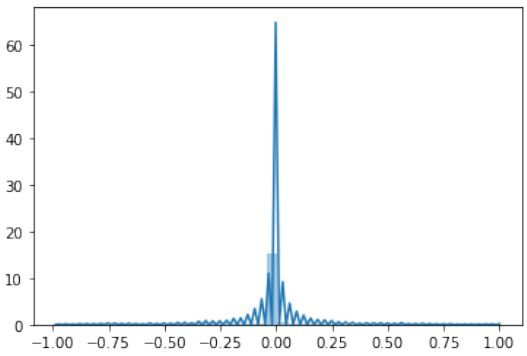


Fig. 1: Original Data Distribution

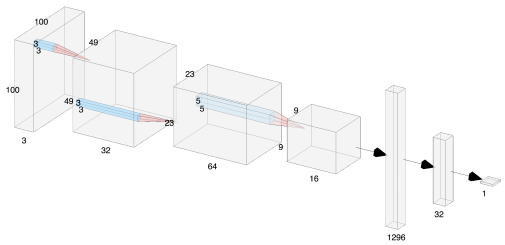


Fig. 2: CNN Model

B. Model Training

We trained a 3 layered CNN Regressor model which can be seen in 2. Our Input to the model was 100 $\times$ 100 $\times$ RGB image while our target variable was the unscaled angular velocity.
Our Initial model did not perform up to the mark due to overfitting. This problem was resolved by using L2 regularization and adding 2 dropout layers. Using dropout layers helped in stabilising our model in the simulation world by a lot.

The probability distribution of our training data's angular velocity can be seen in the 5

Other than this we also trained a ResNet18 model for comparison of our results. We made some modification on the Resnet18 model. As Resnet18 model is trained on ImageNet dataset it gives an output of 1000(Imagenet does classification on 1000 classes) therefore we added two linear layer with ReLU and tanh activation functions to get our unscaled angular velocity. We also froze the weights of the ResNet18 model parameters and only learned the weights of the last two linear layer.

The output range of our unscaled angular velocity(ω) $\in$ [-1,1] When the output ω =1 it means that the obstacle is on the left, while when ω = 0 implies that the robot is symmetric to the sensor or there are no obstacle and finally when ω = -1 it implies that obstacle is on the right.For ω we used the averaging vector of [1,2,0,-2,-1].

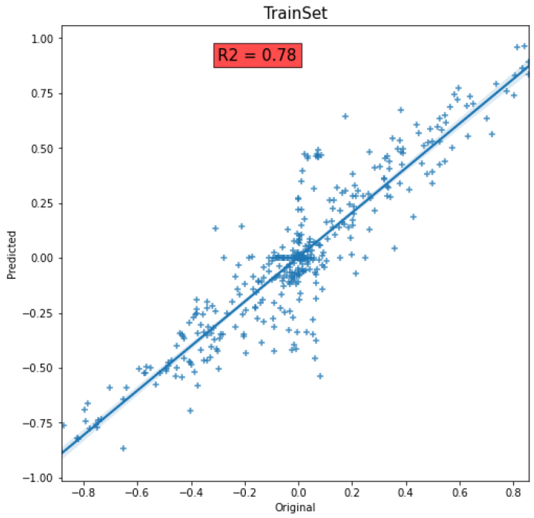
One drawback which the above averaging vector has is that when the robot is perfectly perpendicular to object ahead, the predicted angular velocity would be 0. To overcome this, we trained another model for prediction of centrality for our CNN model. For benchmarking our results, a similar ResNet model was trained for this as well. The output range of our unscaled centrality C $\in$ [-1,1] When the output C =1 it means that the obstacle is centered while when C = 0 signifies that there is no obstacle and finally when C = -1 implies obstacle is not centered. For Centrality we used the averaging vector of [-1,-1,4,-1,-1]

C. Code for ROS

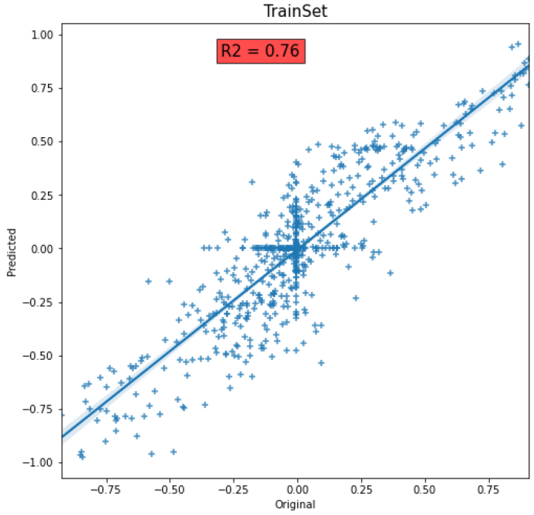
After training our model using the above mentioned CNN regressor we coded the MyT thymio using the ROS package. We used the predicted angular velocity(ω) to avoid any obstacle and the predicted centrality value to identify if our robot is orthonormal to an obstacle. If we found our robot was orthonormal to the any obstacle we stopped our robot and rotated it for some random angle near π.

There were many other changes that were needed for the successful driving of the robot in the simulation world. Some of the major changes were:

1) **Thresholding our angular velocity**: As the CNN model isn't perfect there were instances when there was no obstacle near the robot and our model still predicted angular velocity. To mitigate this problem we used thresholding on our perdicted angular velocity. This threshold helped in moving the robot linearly and using angular velocity when it was near the obstacle. Our threhold value for angular velocity was **0.1**
2) **Rescaling the angular velocity**: As our $\omega \in$ [-1,1], we needed to re-scale this value so as to reduce our overall angular velocity and also easy avoidance of obstacle, for this we multiplied it with a scaling factor of **5**.
3) **Thresholding our Centrality**: As the CNN model isn't perfect there were instances where there were no obstacle near the robot but it still gave some centrality values. To mitigate this problem, we applied a threshold for the predicted centrality value to cross, before it was considered. The threshold value for centrality was **0.2**



(a) Goodness of fit on train set of our CNN model.

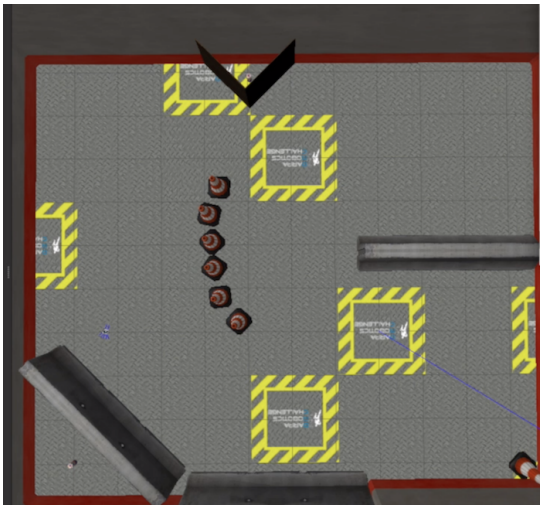


(b) Goodness of fit on train set of our ResNet model.

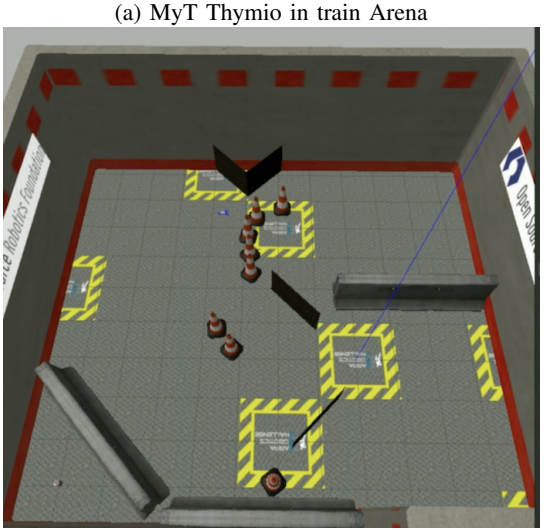
Fig. 3: Goodness of fit on train set.

4) **Multiple High Values of Centrality**: Adding centrality to the controller introduced another problem of over-cautious centrality rotations. This issue could have been minimized by shifting the threshold for centrality but with that, the number of collisions also increased. To solve this problem without increasing the threshold to near 1, we started the centrality sub routine only when our model predicted a value above a threshold for 3 consecutive iterations.

After our model was trained we performed various statistical checks to check the validity of our model and how it might perform in the real scenario for this we computed the R2 value of our model in the train and test for both the modified ResNet18 and our CNN model which can be seen in Figure 3



(a) MyT Thymio in train Arena



(b) MyT Thymio in train Arena.

Fig. 4: Training and Testing arena

D. Simulation in Train/Test set and various metrics

As we received a good R2 on our train and test dataset for both the models we finally decided to run our robot in the Gazebo simulation world. The Thymio robot ran very well in the training world avoiding obstacles majority of the times and using predicted centrality values to identify when it was orthonormal to the wall, followed by stopping and rotating so as to succesfully avoid obstacles. The visuals of our training and testing arena can be seen in the Figure 4

We also ran the robot in the test arena which was slightly different than our training arena. Although the performance wasn't as good as the one observed in train arena it was still pretty good. Moreover from visual inspection it looked like that ResNet18 model was able to generalize better than our CNN model which will also be confirmed in the Experiments section.

After this we came up with various metrics to judge our model and compare our models in the simulation world which are explained in the Experiment subsection.

III. Experiment and Results

: We thought of two different metric to judge the accuracy of our model in the simulation world.

1) **Average collision per min**:
   We ran our thymio robot for 10 min multiple times and computed the average number of collisions per min. We believe that judging our model by number of collision is a good metric. While the CNN model performed better in the train set ResNet18 model was able to generalize much better in the test set. Results of which can be seen and confirmed from Table I
2) **Average distance covered per min**:
   As our Thymio robot had unnecessary rotation even though there were no obstacles around it. To judge this we computed the total distance covered by our robot. A model with high distance covered meant that it was able to successfully avoid obstacle and also minimize unnecessary rotation. In this case again while our CNN model performed better in the trainset, our modification of ResNet18 was able to generalize much better. The result of which can also be confirmed from Table II

The probability distribution of our training data's original and predicted angular velocity can be also be seen in the 5. As can be seen probability distribution of predicted angular velocity is very similar.

TABLE I: Average Collision per min

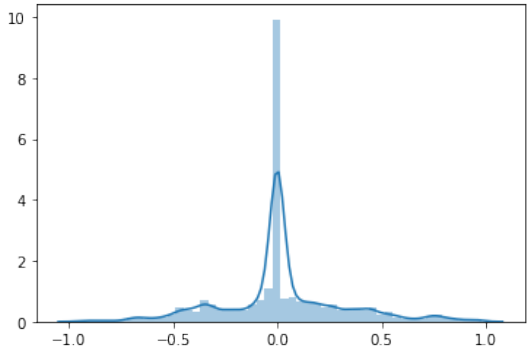
Average Collision per min		
Model	***Trainset***	***Testset***
CNN Model	.8	1.4
ResNet18	.9	1.2

TABLE II: Average distance per min

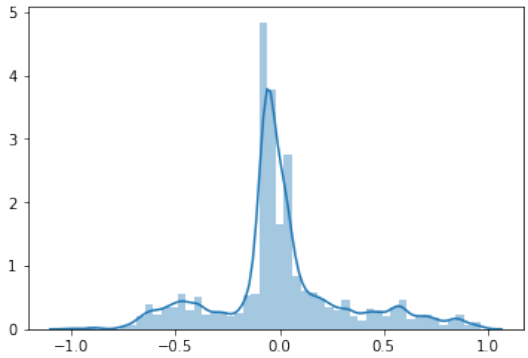
Average distance per min		
Model	***Trainset***	***Testset***
CNN Model	11.4	9.6
ResNet18	9.6	10.4

IV. Future Works

1) We believe that a sequence to sequence model might perform better in predicting upcoming obstacle as it would be able to capture temporal features of navigation and object avoidance. We thought of two seq-2-seq model, first one being a vanilla LSTM in conjuncture with our CNN model and the other one being a ConvolutionalLSTM.
2) We couldn't explore our Hyperparameters because of memory and time constraints so definitely there might be a parametric space where our model performs better.
3) Experimentation in different environment in terms of objects, texture, lighting.



(a) Distribution of Original Unscaled angular velocity



(b) Distribution of Predicted Unscaled angular velocity

Fig. 5: Probablity distribution of our Original and Predicted Angulat velocity

4) Performance of a model trained on a particular arena and testing on an arena completely different visually than the one trained on.

V. Links

1) Github
2) Dataset
3) Model

References

[1] Stanford Artificial Intelligence Laboratory et al, Robotic Operating System, https://www.ros.org ,May,2018.

[2] Paszke, Adam and Gross, Sam and Massa, PyTorch: An Imperative Style, High-Performance Deep Learning Library in Advances in Neural Information Processing Systems, 2019.