COMPARING EFFECTIVENESS OF MACHINE LEARNING VERSUS LOGIC-BASED ALGORITHMS IN TARGET DETECTION

Ang Jin Seng Eugene[1] , Lim Zhi Wen[1], Angel Chia[2], Zhan Yan Jun[2]
[1]Temasek Junior College (Secondary), 22 Bedok South Road, Singapore 469278
[2]DSO National Laboratories, 12 Science Park Drive, Singapore 118225

Abstract
With the rising popularity of machine learning to solve various problems and tasks, this paper aims to see the effectiveness of a machine learning neural network to accurately detect the number of targets in a range-doppler (RD) map, and compare its performance with a traditional logic-based thresholding algorithm. To do this, we programmed a convolutional neural network to classify the number of targets of different RD maps. We also programmed a logic-based algorithm to box up the targets on the RD map and output the number of targets. Thereafter, we compared the accuracy of the neural network with the logic-based algorithm and found that the accuracy of the neural network tended to be higher than the accuracy of the logic-based algorithm. Thus, we concluded that applying neural networks to target detection in radar signal processing could prove to have a higher performance than traditional logic-based algorithms.

1 Introduction
Machine learning and Artificial Intelligence has become the buzzword in the field of science and technology due to its ability to train and teach a machine to imitate intelligent human behaviour. Our project aims to evaluate and compare the effectiveness of a machine learning model as compared to a traditional logic-based algorithm for the target detection portion of radar signal processing.

The machine learning model architecture used in this project is a convolutional neural network, while the logic-based algorithm utilises adaptive thresholding to determine targets. Convolutional neural networks (CNNs) are often used for image classification. By recognizing valuable features, CNNs can identify such features, and classify different images into different classes.

As our project would investigate the effectiveness of the two algorithms, this paper could verify whether machine learning could be better than logic-based algorithms in cases of target detection, which this paper will cover, as well as classification, and perhaps parameter estimation.

As a well designed machine learning model is able to learn and improve over the number of epochs trained, we believe that the convolutional neural network would have a higher accuracy compared to a well designed logic-based algorithm, which would remain at the same accuracy, without tweaking of the algorithm.

2 Methodology
2.1 Data Set Preparation
For our project, the data used was obtained from Github, then accessed and downloaded through a shared Google Drive containing the raw radar data and ground truth data. The data collected is of vehicles and pedestrians on the road and on the side of the road respectively.

The details of the radar configuration used to capture the data is shown below, in Table 2.1.1.

Table 2.1.1: Radar Parameters [1]

Designed Frequency	76.8 Hz
Config Frequency	77 Hz
Range Size	256
Maximum Range	50 m
Doppler Size	256
Azimuth Size	256
Range Resolution	0.195 m/bin
Angular Resolution	0.00614 radian/bin
Velocity Resolution	0.420 (m/s)/bin

The raw radar data was used to plot range velocity maps used as the input data in our project.

With our focus being correctly detecting the number of targets, the ground truth for the correct number of targets was taken as the number of items in the category “classes”.

For this dataset, there are 6 different types of targets, namely person, bicycle, car, motorcycle, bus, truck. An example of the angle-range and range-Doppler maps would be shown below. However, as our project focuses more on the RD maps for detection, we did not use the angle-range but only the range-Doppler maps. Fig 2.1.1 shows an example of the angle range map and RD map.

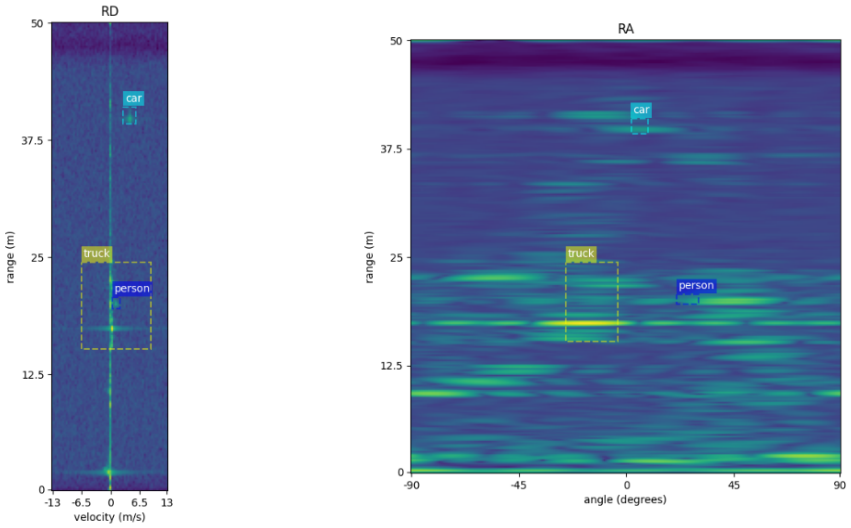


Fig 2.1.1: Example of Range-Doppler and Angle-Range Maps [1]

The data that we had was split into training data and validation data, as well as test data, where the split of data, as well as the number of samples in each dataset will be detailed in Table 2.1.2. Table 2.1.3 will then show the number of samples in each class, where the percentage split between the classes is consistent throughout the 3 datasets.

Table 2.1.2: Dataset Split

	Train	Validation	Test	Total
Percentage Split	71.4% (5/7)	14.3% (1/7)	14.3% (1/7)	100% (7/7)
Number of samples	597	117	117	831

Table 2.1.3: Number of Samples in Each Class

Number of targets	1	2	3	4	5	6	7
Number of samples	63	189	248	179	114	31	7

For the machine learning code, the ground truth was one hot encoded as a label to enable the machine to output the number of targets. An example of a label that is in class 0 (1 target in the scene) in a numpy array format is shown below.

[1. 0. 0. 0. 0. 0. 0.]

This would be inputted into the machine as a "label" for the image, encoding the number of targets in that image, in the format of a numpy array, where using the same example as the one above, it would look like this: [1. 0. 0. 0. 0. 0. 0]. The filenames and the labels were then put into a .csv file to facilitate the creation of the dataset.

2.2 Logic-based Code

To process our images, we have chosen to grayscale the RD maps, to reduce the number of colour channels to 1, rather than 3, in the case of RGB. This would result in pixels having values between 0-255. This would allow us to reduce the amount of computational power needed to perform thresholding, increasing the efficiency of the algorithm.

With our processed images, most contain a dark line in the middle of the image, due to the presence of stationary clutter (e.g. buildings, road signs) in the scene, which could have negatively affected our results, especially with the nature of logic-based programming being unable to learn trends and better identify targets, the clutter being stronger in the RD map would result in less accurate results. Hence, we changed the pixel values of the clutter to 255 so that it would not be detected as a target. Both the RD map with clutter, and the RD map with the removed dark line.

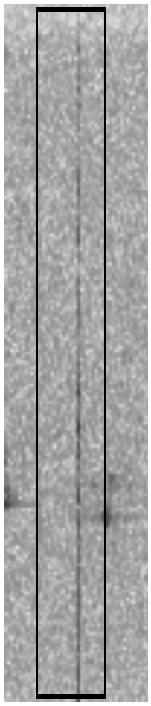


Fig 2.2.1: RD Map with Dark Line

Fig 2.2.2: RD Map with "Removed" Dark Line

For logic-based thresholding, Open Source Computer Vision Library (CV) offers 2 types of in-built thresholding, namely simple thresholding or adaptive thresholding. Adaptive thresholding in CV further offers adaptive mean thresholding or adaptive gaussian thresholding. Adaptive mean thresholding would take the mean of neighbour pixel values minus a user defined constant. Adaptive gaussian thresholding refers to when the threshold value is a gaussian weighted sum of the neighbouring pixel values minus a user defined constant. A more detailed diagram for both adaptive mean and adaptive gaussian would be shown below, in Table 2.2.1.

Table 2.2.1:Details on Adaptive Threshold Methods [2]

Adaptive method	Method used to obtain threshold value
Adaptive threshold mean	the threshold value for a pixel is determined by taking the mean of the pixel values in a window centred around said pixel minus a user-defined constant
Adaptive threshold Gaussian	the threshold value for a pixel is determined by taking is a weighted sum (cross-correlation with a Gaussian window) of the pixel values in a window centred around said pixel minus a user-defined constant
Global Thresholding	The threshold value is a user defined constant

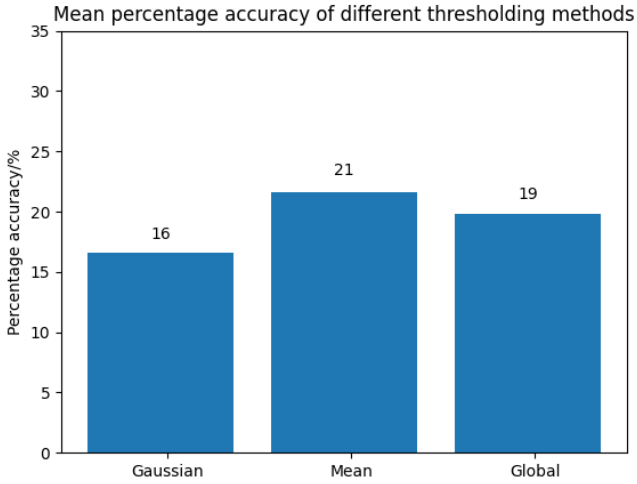


Fig 2.2.3: Bar Graph Displaying Mean Percentage Accuracy of Different Thresholding Methods

The percentage accuracy of each thresholding method is calculated by taking the average percentage accuracy across all images. The percentage accuracy of each image is the number of targets detected in the image over the total number of targets in the scene.

2.3 Machine Learning

To facilitate the identification of the number of targets in the image, we built a convolutional neural network of about 880 thousand programmable params, where we used the "LeakyReLU" activation function, which allows us to prevent dying 'ReLU', where the ReLU neuron will output zero, when the input is of negative values. We also added a training checkpoint function, to ensure that the model is not over-fitted, by saving the model weights when the model has the highest accuracy in the validation set. This allows for more accurate results when the model is tested. We had also included 2 dropout layers, to prevent overfitting. Below, Fig 2.3.1 shows a summary of the model architecture that we used for training.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 349, 72, 3)        0

 conv2d (Conv2D)             (None, 339, 62, 32)       11648

 max_pooling2d (MaxPooling2D  (None, 169, 31, 32)      0
 )

 dropout (Dropout)           (None, 169, 31, 32)       0

 conv2d_1 (Conv2D)           (None, 159, 21, 16)       61968

 max_pooling2d_1 (MaxPooling  (None, 79, 10, 16)       0
 2D)

 dropout_1 (Dropout)         (None, 79, 10, 16)        0

 flatten (Flatten)           (None, 12640)             0

 dense (Dense)               (None, 64)                809024

 dense_1 (Dense)             (None, 7)                 455

=================================================================
Total params: 883,095
Trainable params: 883,095
Non-trainable params: 0
_________________________________________________________________
```

Fig 2.3.1: Machine Learning Model Summary

3 Results

3.1 Logic-Based Algorithm

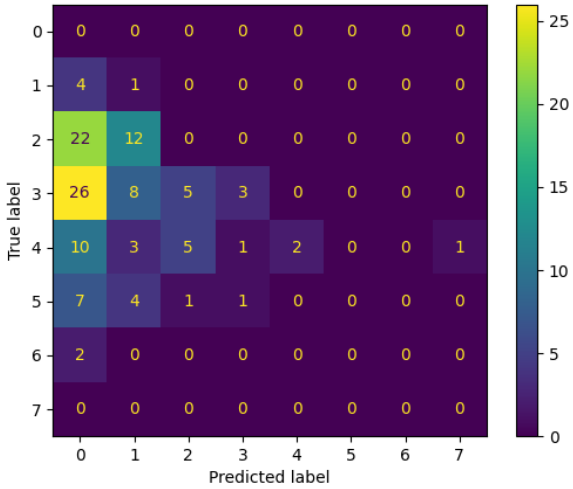


Fig 3.1.1: Confusion Matrix Showing the Predictions of the Logic Algorithm

Our best performing algorithm (Adaptive Mean threshold), tended to detect less targets than what was in the image, as can be seen in Fig 3.1.1, it also had an accuracy of about 5.1% (6/118).

3.2 Machine Learning

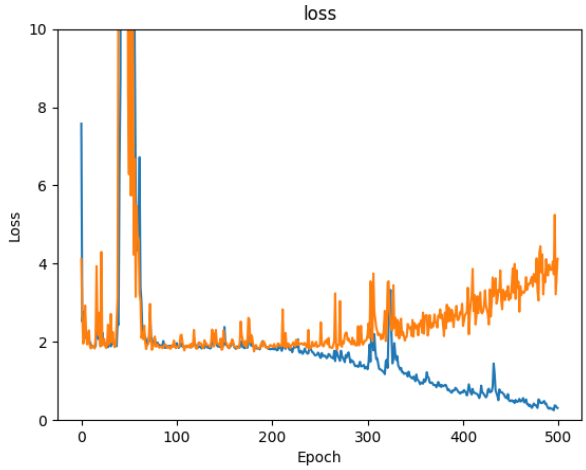


Fig 3.2.1: Graph of Categorical Cross-Entropy loss versus number of epochs

This was our Categorical cross-entropy loss graph while training the machine learning model, where the orange line represents the loss of the validation dataset, and the blue line represents the loss of the training dataset. There are some Epochs where the loss spikes, to about 54 (not graphically shown due to y limit set to 0 to 10) which we would address in the discussion portion of the paper. The training loss tends to decrease with each epoch. However, the validation loss stayed relatively constant from epoch 80 to 230, after which the validation loss increased with each epoch, which suggests overfitting.

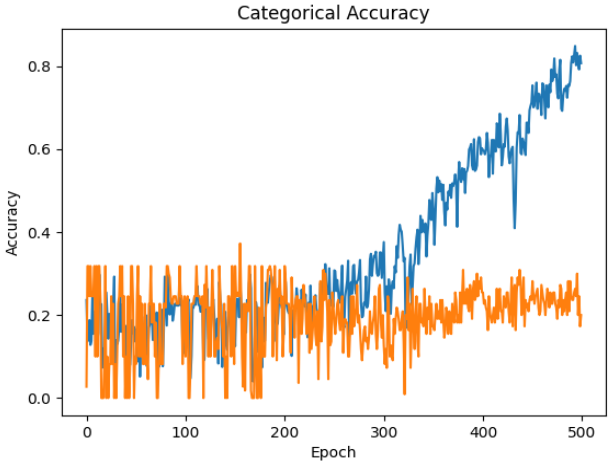


Fig 3.2.2: Graph of Accuracy Over Number of Epochs

With enough training, the training accuracy eventually reaches close to 100%, while the validation accuracy hovers around between 18% to 30%. Notably, the validation accuracy peaks close to 40%, at around epoch 165. This accuracy trend suggests some sort of overfitting.

To evaluate the performance of the model on the test set, we load the weights which yielded the best accuracy on the validation set. The evaluation is shown in the form of the confusion matrix in Fig 3.2.3, and the resultant accuracy is 34.7%.

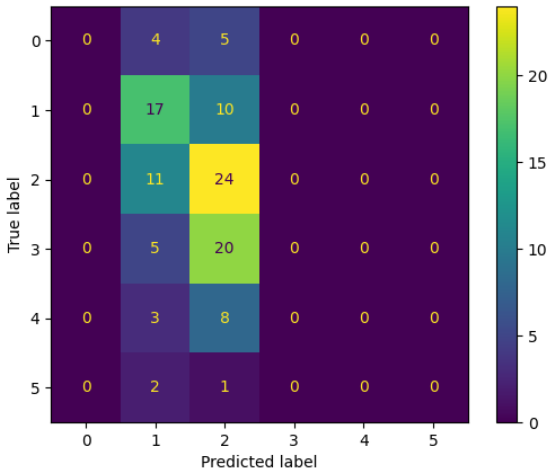


Fig 3.2.3: Confusion Matrix of Test Set for Machine Learning

4 Discussion

4.1 Loss for Machine Learning Algorithm

4.1.1 Loss Spike during Machine training

When compiling the machine before training, we used a tensorflow inbuilt optimizer “Adam”, which is a comprehensive optimizer for machine learning algorithm [3], where it is recommended for use for most machine learning algorithms, due to its superior experimental performance as compared to other optimizers (SGD, RMSP, AdaGrad). However, the loss spike during training could be a consequence of Mini-Batch Gradient Descent with the Adam optimizer, where some batches of data may have unlucky data, by chance [4], which would cause the loss spikes we see in Fig 3.2.1.

4.1.2 Loss while Training the Machine Learning Model

With reference to Fig 3.2.1, our loss graph while training the machine learning model generally decreased with training, from about 3 to 0.145. However, the validation loss is increasing, from 3 to about 5. This signifies some form of overfitting, where the model remembers the training dataset, rather than learning the patterns in it. Although different techniques have been used to combat this problem, such as using class weights, which we calculated using the formula wj=n_samples / (n_classes * n_samplesj), [5] where wj signifies the weight for that class, n_classes signifies the number of classes, and n_samplesj signifies the number of samples in a specific class. We had also included dropout layers to combat overfitting. However, overfitting of our dataset still occurred, and this could be attributed to the imbalance of our data, as well as our small dataset size.

4.2 Comparing Accuracy of Target Detection Between the Machine Learning Model and Logic-Based ALgorithm

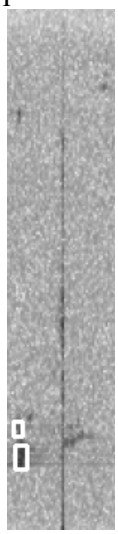
When we compare the accuracy of the machine learning model versus the logic-based thresholding algorithm, against the same dataset size, the machine learning model would have a higher accuracy, of about 34.7%, as compared to logic’s lower accuracy of 21%, when using the adaptive mean thresholding method. Such low accuracies could be in part due to the nature of our dataset, which was very unbalanced, and of a very small size (831 samples in total). This could have affected the performance of our model, where the training of the model on a smaller dataset, could have caused less patterns to be recognised. [6] In the case of the logic algorithm, more thresholding methods could have been explored, to fine tune the algorithm better.

However, with the accuracy of the machine learning model being greater than the logic’s accuracy, from our results, it can be said that the machine learning model would be better suited for target detection. This would be due to the nature of machine learning, where it leverages data to improve performance on this set of tasks, target detection.

4.3 Logic-Based Algorithm Accuracy

While obtaining results for the mean accuracy of each threshold method, we realised that the logic-based algorithm tended to misreport the targets, where the target that was identified by the threshold would not be an actual target in the scene. An example of this can be seen in Fig 2.2.2, where a target is correctly highlighted, whereas the other target that is highlighted is not an actual target in the scene itself.

The logic-based algorithm typically under-detected the number of targets in each picture. This could be due to the fact that the ground truth was taken from the dataset provider, rather than being based off the RD map. The RD maps occasionally had anomalies such as the ground truth being 2 when the RD map had 4 visible targets, or the ground truth being 2 when the RD map had only 1 visible target. This could be due to issues with the data collection or radar hardware problems that detected false targets. An example of this will be shown in Fig 4.3.1.



```
import pandas as pd

groundtruth = pd.read_pickle("/content/drive/MyDrive/groundtruth/000052.pickle")
print("Number of targets in scene: ", str(len(groundtruth["classes"])))
```

```
Number of targets in scene:  4
```

Fig 4.3.1: Image with 2 Targets Detected / Visible, but with 4 Targets in the Groundtruth

5 Conclusions

5.1 Limitations

Some limitations of our project include the type of input data used and dataset limitations. As for the type of input data, we plotted the RD map from the raw data rather than using the raw data, which could have resulted in discrepancies in our findings. For limitations of our dataset, it was greatly imbalanced, as seen in Figure 2.1.3. Class 2, the class with the highest number of samples had 248 samples, while Class 6, the class with the lowest number of samples, had less than 10. Class weights were added to the machine learning algorithm to mitigate this issue, however the imbalanced dataset could have still affected our results greatly. Furthermore, our dataset had a maximum of 7 targets in the scene, whereas during actual deployment more targets can be present in a singular scene, resulting in discrepancy in our findings when it comes to actual deployment.

Secondly, another limitation would be the labelling of our dataset. We took a more unbiased approach to data labelling as the labels came solely from the ground truth, with no reference to the RD map. However, some RD maps would have 2 visible targets, but have a ground truth of 4 targets, such as in Fig 4.3.1. This may have caused discrepancies in the machine learning's training and subsequent predictions on the test dataset, as well as discrepancies in the number of targets detected versus the ground truth.

Lastly, our logic-based algorithm did not take into account the ground truth coordinates of the targets in the images. Hence, the algorithm may have detected clutter or noise as targets, rather than actual targets in the scene. This could result in wrong conclusions as the performance of the

thresholding algorithm was evaluated by taking the mean accuracy of the algorithm across images, without taking into account that the targets detected may not have been actual targets in the scene.

5.2 Future Directions

Future projects could address the data imbalance problem by utilising more data or utilising a dataset captured over a longer period of time, on a busier road, e.g. the highway so as to address the limitation of our dataset imbalance, as well as the limited variety of the number of targets in the scene.

Additionally, future projects could also look into the usage of different types of input data, such as using raw data instead of an RD map. Future projects could also change the calculation of the mean accuracy. Rather than using the number of targets detected to calculate the mean percentage accuracy, the mean percentage accuracy calculation could also take into account whether the detected targets are targets present in the scene.

Furthermore, our project focuses on machine learning applications in target detection, however machine learning applications in the rest of the radar signal processing chain, such as target estimation, identification and classification, could also be further looked into, where machine learning could be used to classify different types of targets.

6 Acknowledgements

We would like to acknowledge the efforts of our mentors, Yan Jun and Angel, for providing guidance and technical support throughout the project. We would also like to thank Chester Tan Wei Jie, who although is not our designated mentor, still provided us with extensive technical support throughout the project.

7 References

[1] Ao Zhang. (2021, May 2). RADDet. GitHub. https://github.com/ZhangAoCanada/RADDet

[2] OpenCV. (n.d.). OpenCV: Miscellaneous Image Transformations. Docs.opencv.org. Retrieved December 30, 2022, from https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#ggaa42a3e6ef26247da787bf34030ed772caf262a01e7a3f112bbab4e8d8e28182dd

[3] Ajagekar, A. (n.d.). Adam - Cornell University Computational Optimization Open Textbook - Optimization Wiki. Optimization.cbe.cornell.edu. Retrieved December 30, 2022, from https://optimization.cbe.cornell.edu/index.php?title=Adam

[4] neural networks - Explanation of Spikes in training loss vs. iterations with Adam Optimizer. (2018, September 21). Cross Validated. https://stats.stackexchange.com/questions/303857/explanation-of-spikes-in-training-loss-vs-iterations-with-adam-optimizer

[5] Singh, K. (2020, October 6). How to Improve Class Imbalance using Class Weights in Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/

[6] Barman, R. (2019). Transfer Learning for Small Dataset Cloud Infrastructure Management View project Classification of text based complaints using NLP and Neural Network View project Anita Patil P K Technical Campus.