Neural Networks for Radar Data Processing

Varsha Nimbaragi[1], Smitha Sasi[2]
[1]*Dayananda Sagar College of Engineering, Bengaluru*
[2] *Dayananda Sagar College of Engineering, Bengaluru*
(varshanimbargi663@gmail.com)

***Abstract*—** A work that radar Processor does is that it collects that data and other parameters of the target and processes that data and work on it. The prediction of that target parameters is done by radar data processor. The parameters that are predicted are position, velocity and acceleration. Distance, pitch angle and azimuth angle is what considered while predicting the position where as velocity and acceleration are considered as motion parameters. The data that is collected by data processor finds it difficult sometimes because the data is complicated. The data that consist of high maneuvering conditions or the data with missing measurements adds the complication. These types of complications can be removed by neural networks. Especially the networks like Long Short-Term memory networks which are also called as Recurrent Neural networks are able to solve the problems. These kinds of networks can be used in the applications like time series forecasting which are able to work with so many numbers of input variables. The objective of this paper is to use the technique of neural networks like Long Short-Term Memory Recurrent Neural Networks to help the radar data processor to deal with the complications and in generation on scenarios and tracking.

***Keywords*—** *Long Short-Term Memory; Recurrent Neural Networks; Benchmark Scenarios.*

I. INTRODUCTION

Nowadays Artificial intelligence is used everywhere from the high end technologies like self driving cars to the low end technologies. It is considered to have many numbers of applications and advancement in the technology has made it to adapt in so many number of applications. Machine Learning and Deep Learning which are part of Artificial Intelligence are having many applications. They consist of many algorithms which are able to solve many of the complication problems. Deep learning is the one that works with neurons and hidden layers which is exactly structured as that of human brain. As human brain consist of dendrites and neurons in the same way artificial neural network is constructed with neurons and the layers that is connecting them together to perform specific operations.

In Radar recently the advancement can be seen where Artificial Intelligence can be adapted. There are many kinds of neural networks which can be adapted in Radar technologies. In this paper the use of Neural Networks like LSTM-RNN can be seen where they are used in radar data processing to remove the complications. The other neural networks fail to solve problem with many input variables and in time forecasting problems. Recurrent Neural Networks are the networks which stores the information in them.[1] The problem that is faced by Recurrent Neural networks can be solved by the Long Short-Term Memory networks. Long Short-Term Memory networks store the information that is lastly learned for longer period of time. Training of the model using LSTM networks is easy. More about deep learning models are studied in [2] to [10].

The 6 benchmark trajectories are discussed in this paper which uses the LSTM models in radar trackers.[11]

The explanation of how to implement the deep learning model can be found in this paper which generates the scenarios with many different maneuvering conditions which consist of the data like position, acceleration and velocity. These scenarios that are generated are found to be helpful in the radar processor. This model is also used to compare the results that are generated with the Kalman filter output. The main application can be found is it helps in coasting whenever the data is not available. These models can be implemented by using Deep learning libraries like TensorFlow and Keras and Python programming is used for implementing.

The LSTM-RNN network can be found in the below diagram. The networks have the loops in them and which stores past information in them. The past information is fed back or fed to the other neural networks. In Long Short-term memory neural networks, the short amount of information can be stored for longer amount of time.

II. IMPLEMENTATION

The methodology adopted for achieving the goal is described below.

A. Training Data Generation

A comprehensive set of scenario data of depicting various target maneuvering conditions such as constant velocity, constant acceleration, level turn, climb, turn with acceleration, turn with climb and turn with climb & acceleration has to be generated first. This will be performed though the implementation of motion models in MATLAB. This data is used to develop the LSTM-RNN model. The training dataset represent a target trajectory, which consists of the target's position, velocity & acceleration components and time.

The following hyper parameters are to be considered during the implementation:

- No. of Epochs

- No. of LSTM Layers
- No. of Neurons per hidden layer
- LSTM memory length
- Batch size
- Learning Rate
- Optimization Algorithm
- Loss function
- Dropout (Yes or No)

The approach followed for developing the LSTM model is described in this section. Python based deep learning library named keras is used for the LSTM implementation. As Keras cannot be used as a standalone framework for the implementation, TensorFlow is used as the backend.

The Integrated Development Environment (IDE) used for the implementation is the Spyder IDE. **Spyder** is a useful integrated development package that can be used in Python software creation process.

A snippet of the training dataset is shown in figure 1. Here, column 1 contains time in seconds, columns 2 to 4 represents targets x, y, z coordinates in km, columns 5 to 7 represents the velocity components in m/s and columns 8 to 10 represents the acceleration components in m/s^2.

	1	2	3	4	5	6	7	8	9	10
1	0	5000	10000	2000	0	100	15	-2.0000	2.6888e-16	0.1000
2	0.1000	5.0000e+03	10010	2.0015e+03	-0.2000	100	15.0100	-2.0000	-0.0040	0.1000
3	0.2000	5.0000e+03	1.0020e+04	2.0030e+03	-0.4000	99.9996	15.0200	-2.0000	-0.0080	0.1000
4	0.3000	4.9999e+03	1.0030e+04	2.0045e+03	-0.6000	99.9988	15.0300	-2.0000	-0.0120	0.1000
5	0.4000	4.9998e+03	1.0040e+04	2.0060e+03	-0.8000	99.9976	15.0400	-1.9999	-0.0160	0.1000
6	0.5000	4.9998e+03	1.0050e+04	2.0075e+03	-1.0000	99.9960	15.0500	-1.9999	-0.0200	0.1000
7	0.6000	4.9996e+03	1.0060e+04	2.0090e+03	-1.2000	99.9940	15.0600	-1.9999	-0.0240	0.1000
8	0.7000	4.9995e+03	1.0070e+04	2.0105e+03	-1.4000	99.9916	15.0700	-1.9998	-0.0280	0.1000
9	0.8000	4.9994e+03	1.0080e+04	2.0120e+03	-1.5999	99.9888	15.0800	-1.9997	-0.0320	0.1000
10	0.9000	4.9992e+03	1.0090e+04	2.0135e+03	-1.7999	99.9856	15.0900	-1.9997	-0.0360	0.1000
11	1	4.9990e+03	1.0100e+04	2.0151e+03	-1.9999	99.9820	15.1000	-1.9996	-0.0400	0.1000
12	1.1000	4.9988e+03	1.0110e+04	2.0166e+03	-2.1998	99.9780	15.1100	-1.9995	-0.0440	0.1000
13	1.2000	4.9986e+03	1.0120e+04	2.0181e+03	-2.3998	99.9736	15.1200	-1.9994	-0.0480	0.1000
14	1.3000	4.9983e+03	1.0130e+04	2.0196e+03	-2.5997	99.9688	15.1300	-1.9993	-0.0520	0.1000
15	1.4000	4.9980e+03	1.0140e+04	2.0211e+03	-2.7997	99.9636	15.1400	-1.9992	-0.0560	0.1000
16	1.5000	4.9978e+03	1.0150e+04	2.0226e+03	-2.9996	99.9580	15.1500	-1.9991	-0.0600	0.1000
17	1.6000	4.9974e+03	1.0160e+04	2.0241e+03	-3.1995	99.9520	15.1600	-1.9990	-0.0640	0.1000
18	1.7000	4.9971e+03	1.0170e+04	2.0256e+03	-3.3994	99.9456	15.1700	-1.9988	-0.0680	0.1000
19	1.8000	4.9968e+03	1.0180e+04	2.0272e+03	-3.5993	99.9388	15.1800	-1.9987	-0.0720	0.1000
20	1.9000	4.9964e+03	1.0190e+04	2.0287e+03	-3.7992	99.9316	15.1900	-1.9986	-0.0760	0.1000
21	2	4.9960e+03	1.0200e+04	2.0302e+03	-3.9990	99.9240	15.2000	-1.9984	-0.0800	0.1000
22	2.1000	4.9956e+03	1.0210e+04	2.0317e+03	-4.1989	99.9160	15.2100	-1.9982	-0.0840	0.1000
23	2.2000	4.9952e+03	1.0220e+04	2.0332e+03	-4.3987	99.9076	15.2200	-1.9981	-0.0880	0.1000
24	2.3000	4.9947e+03	1.0230e+04	2.0348e+03	-4.5985	99.8988	15.2300	-1.9979	-0.0920	0.1000

Fig.1. Dataset

B. Load Dataset

The first step is to define the number of features in the dataset, number of values to be memorized (represented by the variable 'lookback') by the LSTM and the percentage of data to be used for training. The lookback is set as 500 and 70% of the data is used for training and the rest for testing.

And next step is to load the dataset into memory. The data file is loaded into memory using the Pandas read_csv() function. Here, the next step is to prepare the dataset for the LSTM. This involves framing the dataset as a supervised learning problem and normalizing the input variables. All the features are normalized using a min-max scalar to fit into the range [0, 1], and then the dataset is transformed into a supervised learning problem.

C. Define and Train the Model

In this section, fit an LSTM on the multivariate input data. First, split the prepared dataset into train and test sets, then splits the train and test sets into input and output variables. Finally, the inputs (X) are reshaped into the 3D format expected by LSTMs, namely [no. of samples, lookback, no. of features].

Now LSTM model can be defined and fit it for the training data. The model definition varies as the no. of layers and the no. of neurons per layer are hyper parameters. Different values have to be tried to fine-tune these parameters to arrive at the optimum number of layers and number of neurons. A 3-hidden layer LSTM model with 150 neurons in the first hidden layer, 150 neurons in the second hidden layer and 100 neurons in the last layer is considered here. The output layer has 9 neurons representing the 9 features in the scenario. Mean Absolute Error (MAE) loss function with the efficient "*RMSprop"* version of stochastic gradient descent is used in this implementation.

III. EVALUATION

A. Training

The model is then trained for 40 epochs with a batch size of 32. During this training, 30% of the training data is used for validation. Callbacks are used to get a view on internal states and statistics of the model during training. Callback is a set of functions to be applied at given steps of the training procedure.

A trained model will be used without having to retain it on pick-up training where it is left off in case the training process was interrupted. The tf.keras.callbacks.ModelCheckpoint call back allows to repeatedly saving the model both during and also the end of the training. The model is saved only if there's an improvement within the validation accuracy.

An LSTM-RNN model, which can accept past states of targets and predict their future states, is created. Different models including variants of LSTM-RNN will be developed so that the optimal model can be chosen for the final implementation.

B. Evaluate Model

After the model is fit, the entire test dataset can be forecasted. The trained network model is loaded first and the test dataset is given as an input to this model to forecast the rest of the scenario. The forecasted output is then scaled back to original scale and saved. With forecasts and actual values in their original scale, accuracy and error score for the model can be calculated. The loss and accuracy are then plotted.

```
-------
Epoch 00039: val_acc did not improve from 0.99716
Epoch 40/40
```

```
12320/12320 [==============================] - 1815s 147ms/step - loss: 3.9384e-05 - acc: 0.9960 - val_loss: 1.3530e-05 - val_acc: 0.9977

Epoch 00040: val_acc improved from 0.99716 to 0.99773, saving model to Scenario_Predictions_Scaled_Circular_3Features_1500U_100LB_Noisy.hdf5 dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

Plots showing the model accuracy & model loss during training & validation are shown below. Once the training phase was completed, the trained LSTM model with best validation accuracy got saved in to a hierarchical data format (.hd5).

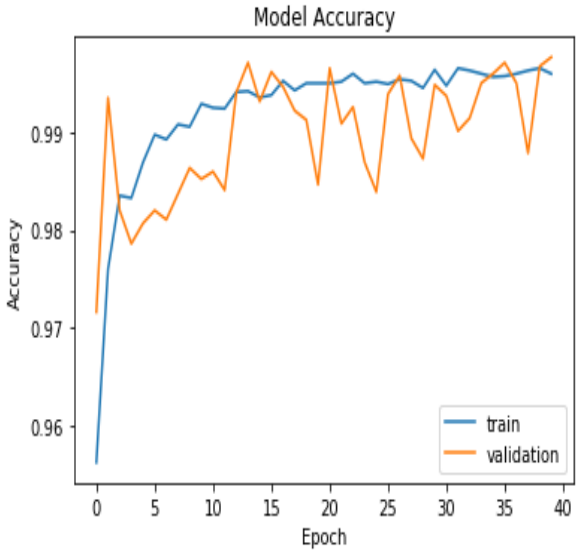


Fig.2 Accuracy

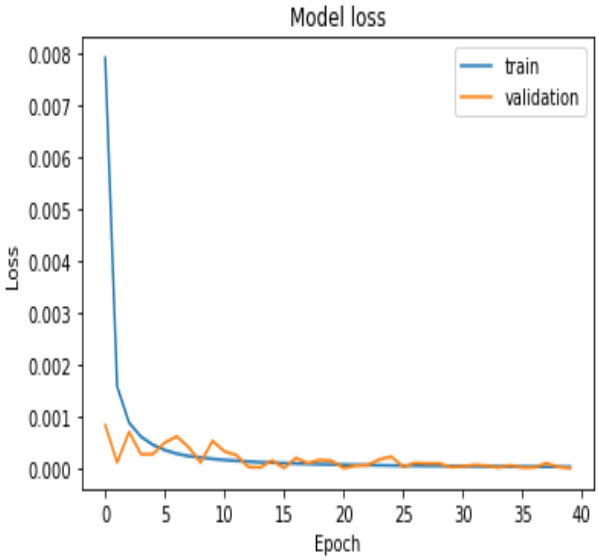


Fig.3 Loss Plot

The performances of the LSTM model were compared against that of a 3-model Kalman Filter based IMM tracker. The prediction output for the last 30% of the unused training data is shown in below figures.

IV. RESULTS

Blaire, et. al, had introduced 6 benchmarking trajectories in their paper. These benchmarking problems are now widely used for benchmarking Radar Trackers.

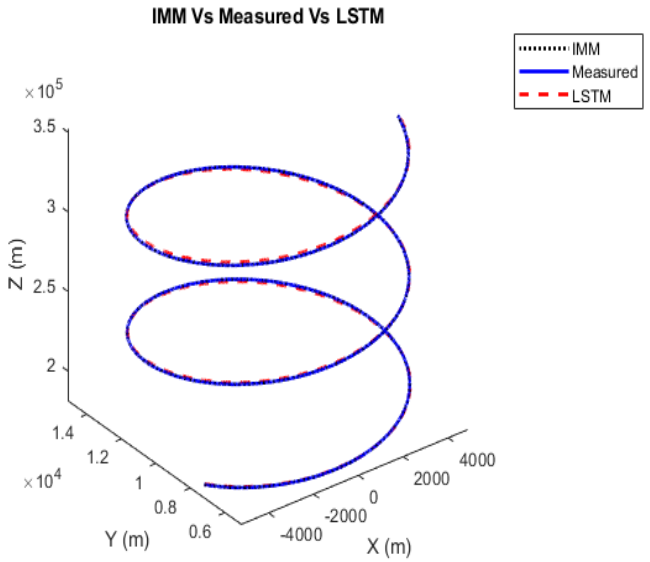


Fig.4. IMM Vs Measured Vs LSTM

Few outputs for LSTM and IMM are shown with benchmark scenarios.

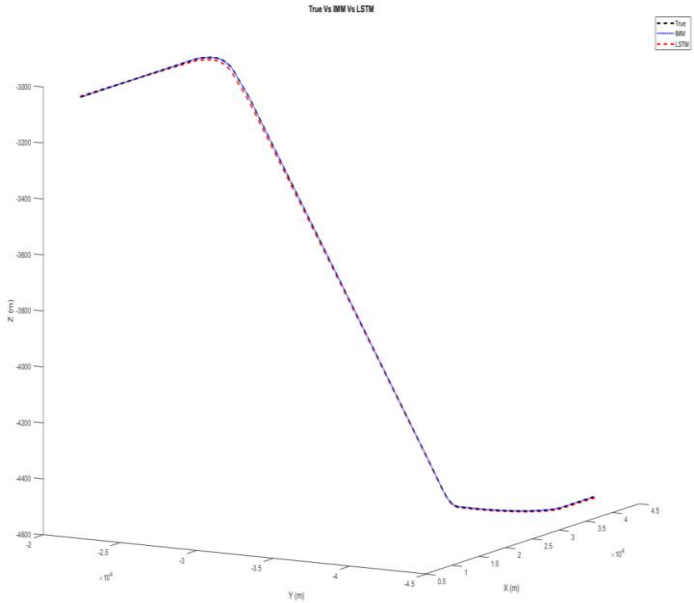


Fig.5 Comparison of LSTM Results with IMM Results & True Scenario for Benchmark 2

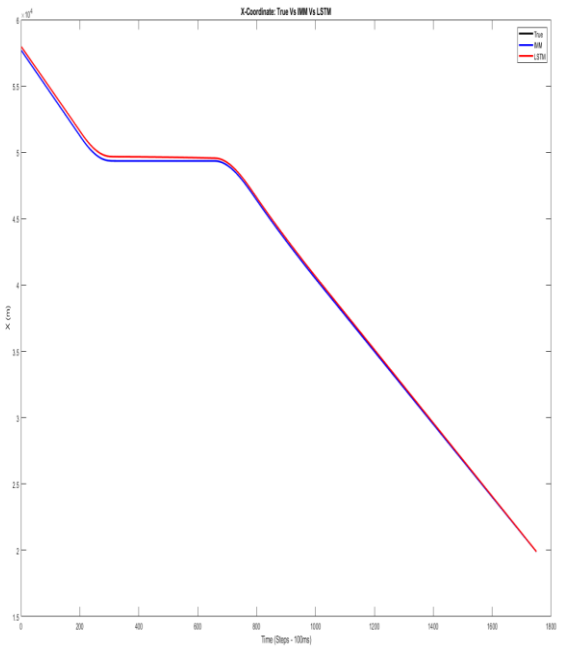


Fig. 6.Comparison of LSTM Results with IMM Results & True Scenario for Benchmark 3 (x-coordinate)

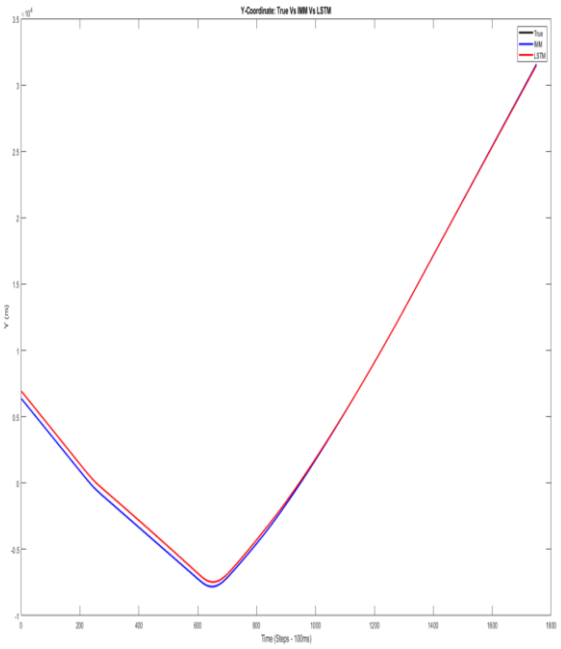


Fig. 7. Comparison of LSTM Results with IMM Results & True Scenario for Benchmark 4 (y-coordinate)

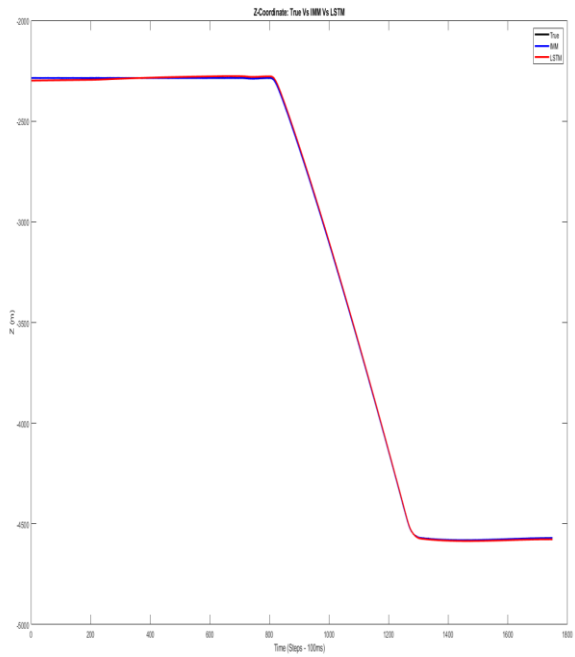


Fig. 8. Comparison of LSTM Results with IMM Results & True Scenario for Benchmark 4 (z-coord)

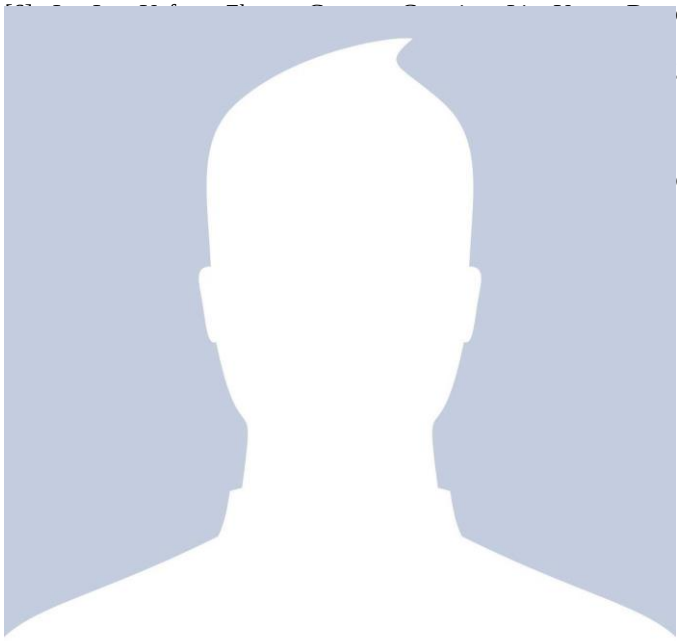
When compared with all 6 cases it can be seen that LSTM model provides good results that helps in tracking.

V. CONCLUSION

The objective of this work is to explore the feasibility of using deep learning in Radar tracking. The results obtained with the benchmark trajectories are very much promising and are as good as the results of a 3-model IMM Kalman tracker. As no trace of any previous work could be found in the literature, I can proudly say that this is very first study on the usage of AI techniques in radar target tracking. The results can further be improved by adding velocity components, acceleration components and target heading as training inputs. But this will add more complexity to the LSTM model.

References

[1] Christopher Olah, Understanding LSTM Networks, https://colah.github.io/posts/2015-08-Understanding-LSTMs/, August 27, 2015.
[2] Bakker, Indra den, Python Deep Learning Cookbook, Packt Publishing Ltd, 2017.
[3] Francois chollet, Deep Learning With Python, Manning Publications, 2018.
[4] Ian Good fellow, Yoshua Bengio, Aaron Courville, Deep Learning, The MIT Press Cambridge, Massachusetts London, England, 2016.
[5] J.Patterson,A.Gibson, Deep Learning: A Practitioner's Approach, O'Reilly Media, 2017.



[9] Nishant Shukla, Kenneth Fricklas, Machine Learning With TensorFlow, Manning Publications Co, 2017.
[10] Phil Kim, MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence, APRESS, 2017.
[11] W.D. Blair, G. A. Watson, T. Kirubarajan, Y. Bar-Shalom, "Benchmark for Radar Allocation and Tracking in ECM." Aerospace and Electronic Systems IEEE Trans on, vol. 34. no. 4. 1998.
[12] https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras